AD-A096 092 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL DCA100-79-C-0037
UNCLASSIFIED BBN-4565-VOL-2 NL

1 OF 5

AD A
096092

Bolt Beranek and Newman Inc.

AD A096092

LEVEL

(12)

Report No. 4565

Design and Real-Time Implementation of a Robust APC Coder for Speech Transmission Over 16 Kb/s Noisy Channels.

Volume II: Real-Time Implementation.

Final Report.

DTIC
MAR 9 1981

December 1980

Prepared for:
Defense Communications Agency

DISTRIBUTION STATEMENT A

DDC FILE COPY

81 3 09 075

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN Report No. 4565, Vol. II	2. GOVT ACCESSION NO.: AD-A096 092
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR SPEECH TRANSMISSION OVER 16 Kb/s NOISY CHANNELS Vol. II: Real-Time Implementation	5. TYPE OF REPORT & PERIOD COVERED: Final Report July 1979 - December 1980
	6. PERFORMING ORG. REPORT NUMBER: BBN Report No. 4565
7. AUTHOR(s): J.J. Wolf, K.D. Field, W.H. Russell	8. CONTRACT OR GRANT NUMBER(s): DCA100-79-C-0037
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, MA 02238	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305	12. REPORT DATE: December 1980
	13. NUMBER OF PAGES: 417
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Defense, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech coding, 16 kb/s speech transmission, adaptive predictive coding, digital voice terminal, real-time speech coder, medium-bandwidth speech transmission, array processor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design and development of a real-time robust APC speech coder that transmits high-quality speech over a 16 kb/s channel with bit-error rates of up to 1%. This volume (Vol. II) describes the implementation of the optimized adaptive predictive coding algorithm as a full-duplex real-time speech coder on the CSP Inc. MAP-300 array processing computer. The report documents in detail the MAP-300 program, and it includes complete listings of the real-time coder program.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4565

DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR SPEECH TRANSMISSION OVER 16 Kb/s NOISY CHANNELS

Final Report

Volume II: Real-Time Implementation

Authors: J. Wolf, K. Field, W. Russell

December 1980

Distribution/
Availability Codes
Dist
A

Prepared for:
Defense Communications Agency

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The purpose of this project was the design and development of a real-time speech coding system that transmits high-quality speech at a data rate of 16 kb/s (kilobits/second). The final report of this project is organized in two volumes. Volume I describes our work in the development and optimization of the speech coding algorithm. Volume II (this volume) describes the implementation of the final optimized speech coding algorithm as a real-time full duplex system on a CSP Inc. (CSPI) MAP-300 signal processing computer and associated hardware.

In the body of this volume we present detailed documentation of the MAP-300 real-time implementation of the speech coding algorithm (Section 2) and instructions on the installation and use of the speech coder software (Section 3). Contained in appendices are function descriptions of BBN-written MAP-300 modules (Appendix A), documentation of the MAP-300 buffers and scalars (Appendices B and C), and listings of the MAP-300 and PDP-11 (FORTRAN) programs that constitute the speech coder system (Appendix D).

2. REAL-TIME IMPLEMENTATION

2.1 OVERVIEW

This section describes the function, components, and design of the BBN16 real-time speech coder implementation. The later sections in this chapter describe in more detail the operation of the speech coder system, the hardware and software used to construct it, and the real-time performance of the completed system.

2.1.1 System Function

The speech coder system is a full duplex terminal of a complete communication system. It is intended to be connected via a 16 kb/s digital I/O link through a communication channel to an identical system. The speech coder system functions simultaneously as both a transmitter and a receiver.

2.1.2 System Components

The speech coder system contains both hardware and software elements. The hardware elements include a CSP Inc. MAP-300 array processor attached to a PDP-11, a handset including microphone and earphone, a hookswitch, amplifiers and low-pass filters, and digital line interfaces. The software elements include MAP-300 programs, which comprise the real-time software, and the program that runs on the PDP-11, which is used only to load and

initialize the MAP software. Fig. 1 is a block diagram of the system.

2.1.3 System Design

The real-time speech coder consists of six separate foreground processes. The Transmitter requires an analog-in process, an analysis process, and a digital-out process. The Receiver requires a digital-in process, a synthesis process, and an analog-out process. Since all of these processes make use of the MAP-300 CSPU to some extent, a mechanism for scheduling them is necessary. Part of this mechanism is contained in the MAP hardware interrupt structures and the SNAP-II executive program. The rest is implemented using flags in conjunction with a background process running in the CSPU. Fig. 2 is a diagram of these processes. The processes share data buffers and communicate the status of these buffers through flags. Since the four I/O processes are continuous, pairs of double buffers are used to enable the necessary sharing.

Each I/O process includes an interrupt service routine, which handles the flags and transfers data to or from the shared buffers. Two levels of double buffering are provided within each I/O process to maintain acceptable system performance under real-time exception conditions.

The background process is shown in Fig. 3. The ANALYZE and

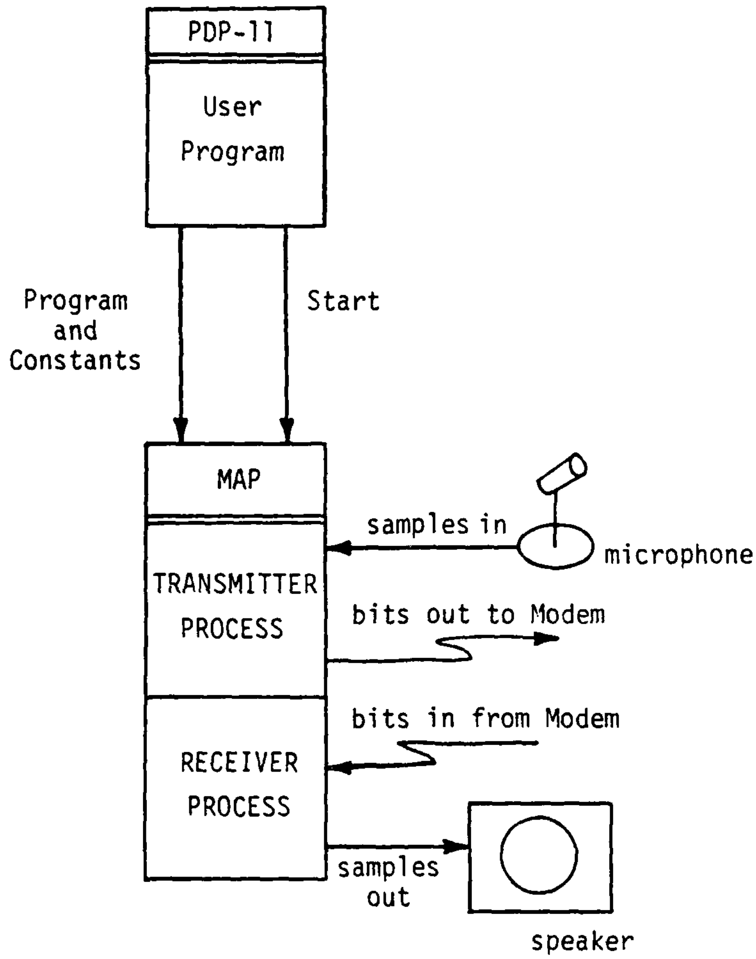


FIG. 1. SYSTEM COMPONENTS

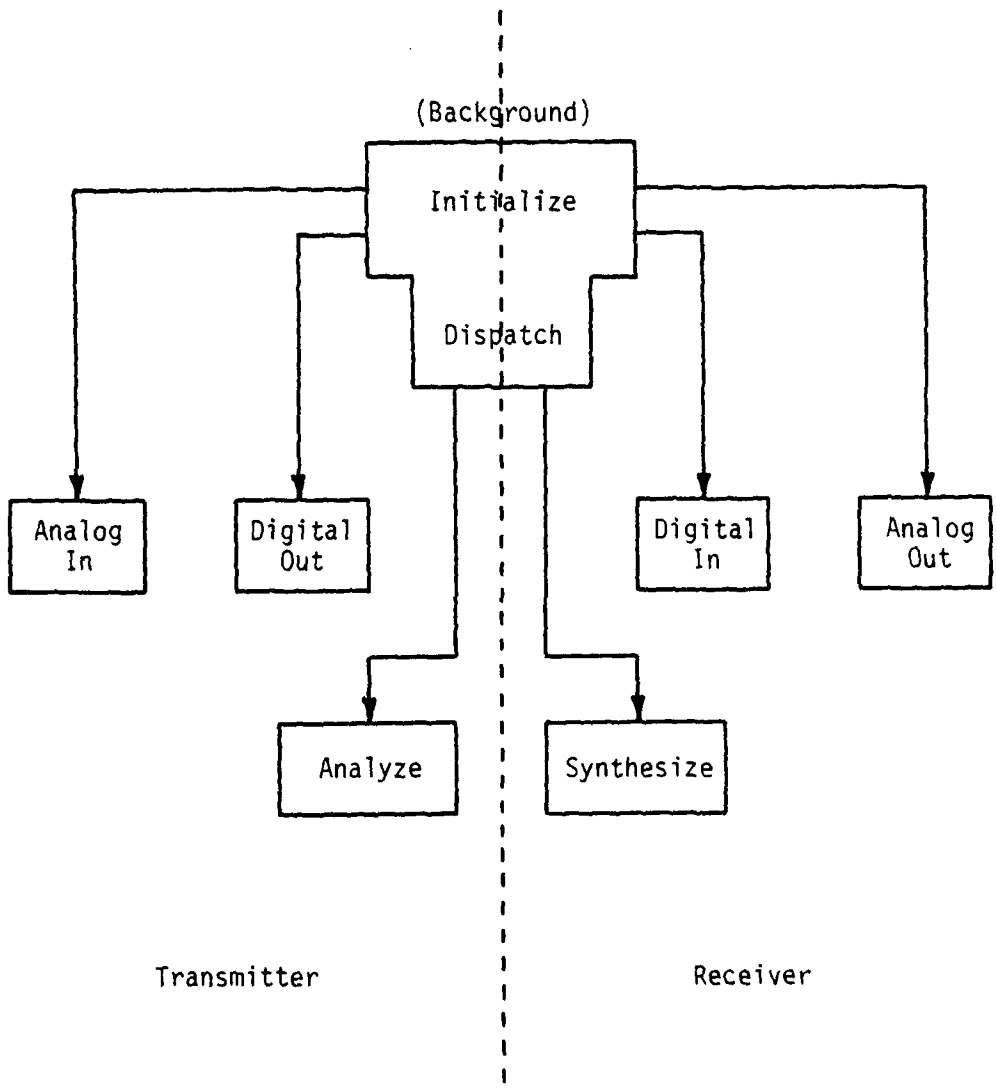


FIG. 2. SYSTEM STRUCTURE

SYNTHESIZE modules are logically asynchronous; they respectively belong to the independent Transmitter and Receiver. However, the modules must be controlled by a sequential machine, the CSPU. The control strategy for allowing each module as much flexibility as possible results in a structure that executes a module only if that module's input and output buffers are available.

The Initialization module is the first module executed, and it is executed only once. It sets all buffer flags to indicate empty, loads and starts the programs in the various I/O Scroll processors, and enables interrupts from these processors.

Control then passes to the basic loop of the background process. This loop executes the ANALYZE and SYNTHESIZE modules when the required I/O buffers are available. For example, the SYNTHESIZEA module will be executed only when the RBITSB buffer is full and the RSINKA buffer is empty.

The ANALYZEA and ANALYZEB modules are functionally equivalent, differing only in their input and output buffers. The structure of the SNAP-II programming environment (specifically, the use of prebound functions for run-time efficiency) does not allow changing the buffers used in a particular function. Therefore, two separate modules have been created to deal with the two sets of TSOURCE and TBITS buffers, A and B. Similarly, two separate SYNTHESIZE modules are required.

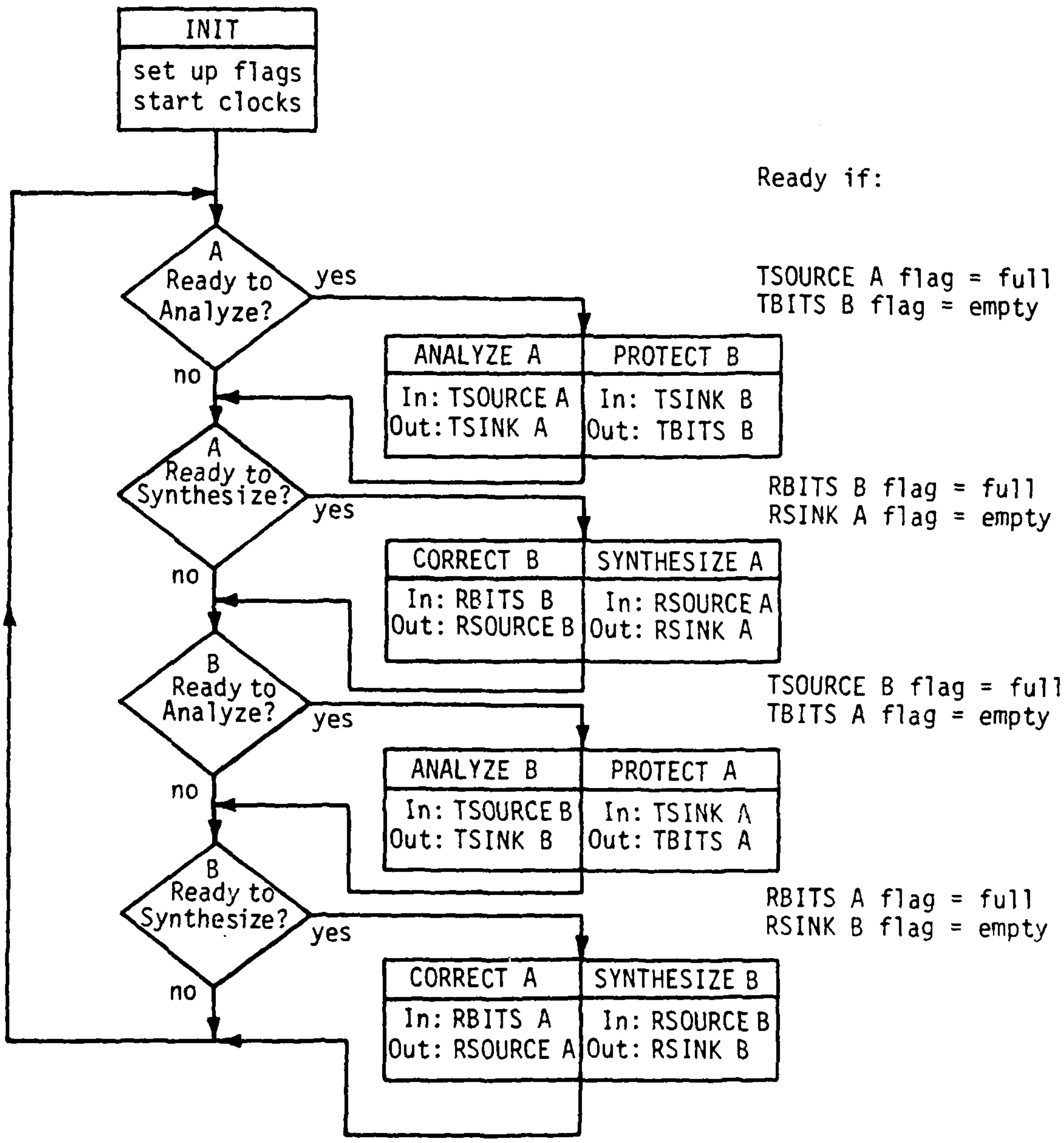


FIG. 3. BACKGROUND PROCESS LOOP

All of the foreground processes are loosely coupled. Execution of neither the ANALYZE nor the SYNTHESIZE module is connected rigidly to any I/O process.

2.2 SYSTEM OPERATION

The speech coder system functions as one terminal of a full duplex digital voice connection. It accepts voice input, digitizes it, and processes it. The processed speech is transmitted as a sequence of bits to a similar terminal. The system also accepts a sequence of bits representing speech transmitted from a remote terminal and processes this sequence to obtain synthetic voice output.

2.2.1 Transmitter

The Transmitter is shown in Fig. 4. The low-pass filtered input speech is fed into an A/D converter contained in an I/O Scroll processor (the ADAM, or Analog Data Acquisition Module), a component processor of the MAP. The program running in this scroll controls the sampling of the speech data, transferring 216 samples (one frame) into each of two buffers, alternately. The sampling rate is 6.621 kHz, so each frame is about 32.625 ms long.

When the ADAM fills one of these buffers, it generates an interrupt to the main MAP processor, the CSPU. The A/D Interrupt

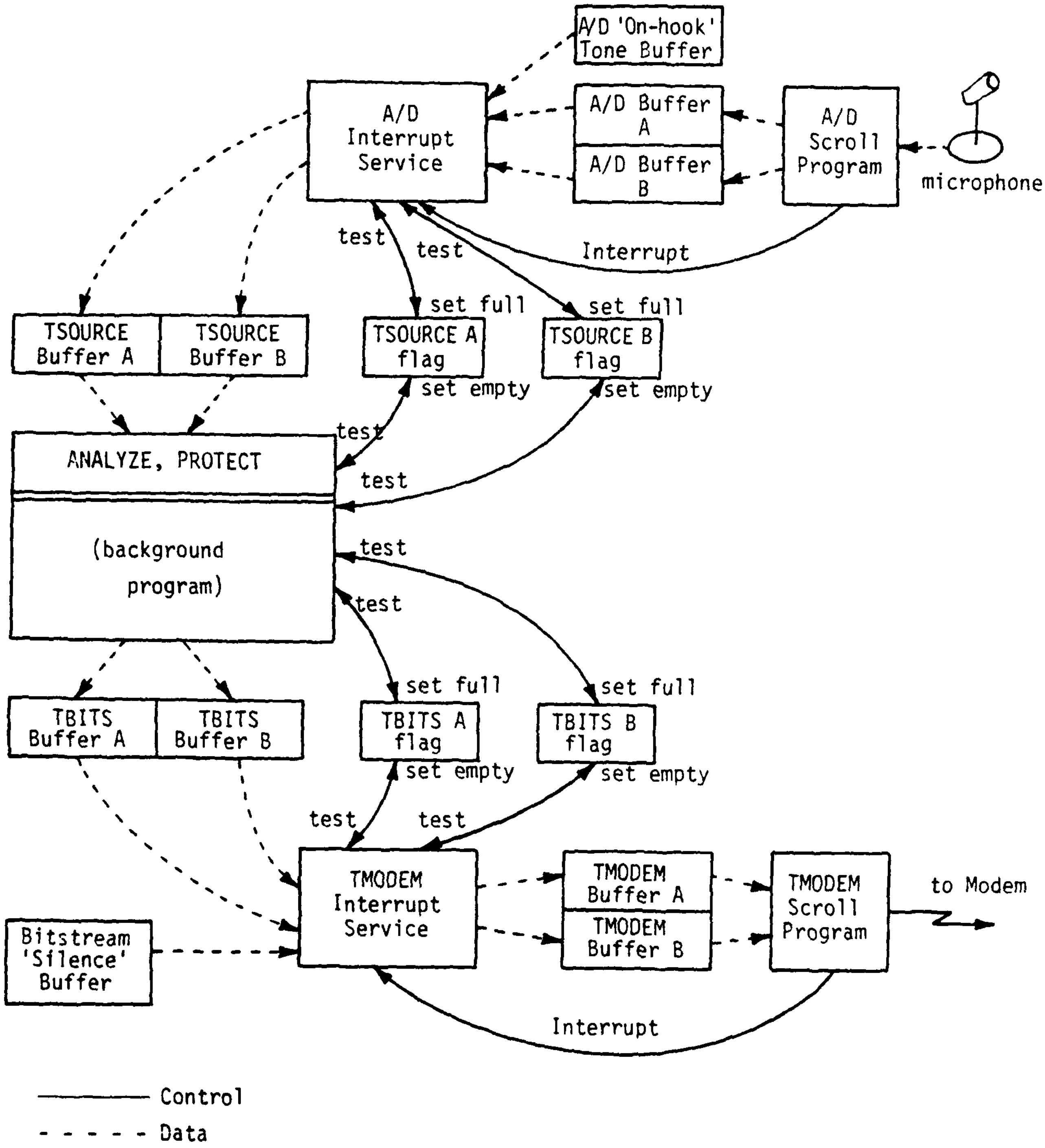


FIG. 4. TRANSMITTER PROCESS

Service routine, which is activated by this interrupt, transfers the data from the just-filled A/D buffer to the current one of the two TSOURCE buffers (if the current one is empty) and sets the corresponding TSOURCEA or TSOURCEB flag to indicate that the buffer is now full. If the current TSOURCE Buffer is not empty, the A/D buffer contents are discarded.

The data in the TSOURCE buffers is used as input by the ANALYZE program module. The Background Process (Section 2.1.3) tests the TSOURCE flags to determine when data is available to the ANALYZE module. After processing a TSOURCE buffer, ANALYZE clears the corresponding flag to indicate that the buffer is empty. The ANALYZE module processes the data, producing coded speech, which is then written into one of the (empty) TSINK buffers. The ANALYZE module runs in the CSPU and AP at background level and is therefore asynchronous with the interrupt service routines.

After ANALYZE fills a TSINK buffer, the PROPAR and PRORES modules transform the data from this buffer into an error-protected bitstream. This bitstream is stored in one of the (empty) TBITS buffers, and the corresponding TBITS flag is set to indicate full. The PROPAR/PRORES operation logically follows ANALYZE, in that data must be processed by ANALYZE before it is available to PROPAR/PRORES. However, the PROPAR and PRORES modules are actually executed concurrently with the next

execution of the ANALYZE module. In fact, they appear as part of the ANALYZE function list. The first time the ANALYZE and PROPAR/PRORES modules are executed, PROPAR/PRORES will operate on a TSINK buffer before it has been filled by ANALYZE; therefore, the TSINK buffers are initialized to coded silence.

The data processed by the PROPAR/PRORES modules will eventually be output to the modem by the TMODEM program, running in another I/O Scroll processor. This program takes data, in the form of a bitstream, from the TMODEM buffers and puts it out to the modem. When the TMODEM buffer is emptied, the TMODEM scroll program generates an interrupt to the CSPU and begins taking data from the other TMODEM buffer.

This interrupt causes execution of the TMODEM Interrupt Service routine. This routine transfers data from a full TBITS buffer to the just-emptied TMODEM buffer and clears the corresponding TBITS flag to indicate empty. If the TBITS buffer is not full, a buffer corresponding to coded silence is transferred to the TMODEM buffer.

2.2.1.1 ADAM Scroll Program (ADPROG)

Speech input is performed by the ADAM, an IOS-2 scroll processor that contains an analog multiplexer, sample/hold, and A/D converter. The ADAM receives both the input speech signal and the input sampling clock from the Speech Processor Interface (SPI). The signal is in the range -5 to +5 volts, and the sampling rate is 6.621 kHz.

The ADAM program (ADPROG) is shown in Fig. 5. The speech samples are double-buffered in A/D input buffers named TADBA and TADBB. (All MAP buffers and most scalars are prefixed with "T" or "R" to indicate that they are used in the speech coder Transmitter or Receiver respectively.) The A/D input buffers are 216 samples long, which corresponds to a frame length of about 32.625 ms. The A/D samples are written into memory in short (16-bit) floating point format. (The ADAM and AOM are capable of working only in 16-bit floating or 16-bit fixed point formats.)

ADPROG sets the ADAM multiplexer to Channel 1 and sets the F1 flag to initiate sampling. ADPROG maintains a pointer into the A/D input buffer, which is initialized to TADBA-1. When a sample is converted, this address pointer is incremented and used to transfer the A/D sample to MAP memory. The address pointer is compared to the buffer end address; when the end has been reached, a line 1 interrupt is passed to the CSPU to signal the filling of the buffer, and the SYNCSTOP register is checked. SYNCSTOP is a register that is set by the CSPU to signal the ADAM to stop sampling. In normal operation, it remains clear, so ADPROG resets the address pointer to the other A/D input buffer (TADBB), and speech sample input proceeds without interruption. Interrupt 1 from the ADAM is used to signal the filling of each successive A/D buffer and the switch to the other A/D buffer.

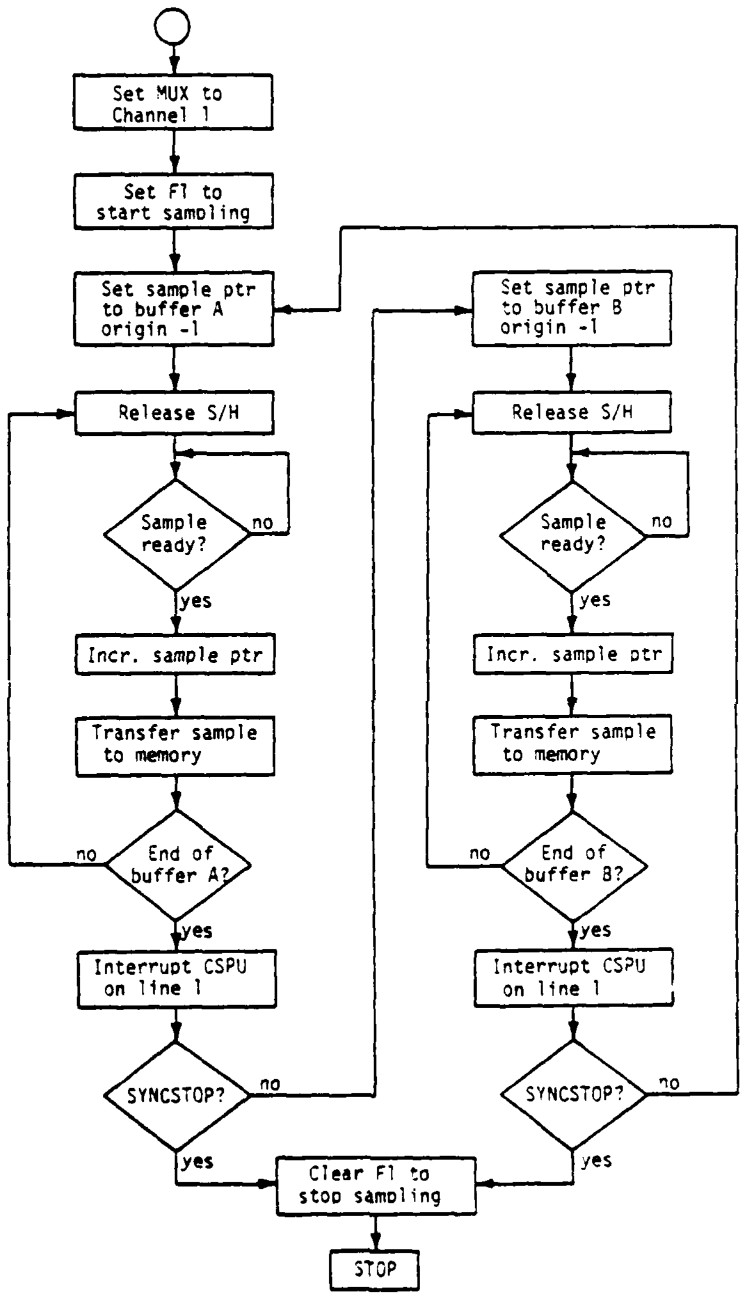


FIG. 5. ADPROG: ADAM SCROLL PROGRAM

2.2.1.2 A/D Interrupt Service (ADAMINT)

ADAMINT is activated by each ADAM line 1 interrupt, signifying the filling of another A/D input buffer. ADAMINT, like the other three speech coder input/output interrupt service routines, maintains a pair of integer scalar "pointer offsets" to keep track of its input and output buffer relationships. These pointer offsets, which take on the values -2 and 0, are used to reference small tables of address pointers. Thus, for example, ADPO ("A/D Pointer Offset") references ADBPTR, a table of pointers to the two A/D input buffers, and TSRPO ("TSOURCE Pointer Offset") references the tables TSRBPTR and TSRFPTR, which point to TSOURCE buffer-copying subroutines and buffer-status flags.

The operation of ADAMINT is shown in Fig. 6. When it receives a buffer-filled (line 1) interrupt, ADAMINT switches ADPO to the buffer just filled by the ADAM.Then ADAMINT checks the status of the next TSOURCE buffer to be filled; if it is available (empty) and the handset hookswitch is up (indicating that the handset is in service), ADAMINT copies the new A/D buffer to the current TSOURCE buffer. If the TSOURCE buffer is empty but the hookswitch is down, a preconstructed "tone" buffer is copied instead to the TSOURCE buffer. (This simulated input is intended to indicate to the person using the remote speech coder that this coder is in operation, but the handset is resting in the holder.) In either case, the TSOURCE buffer flag is set

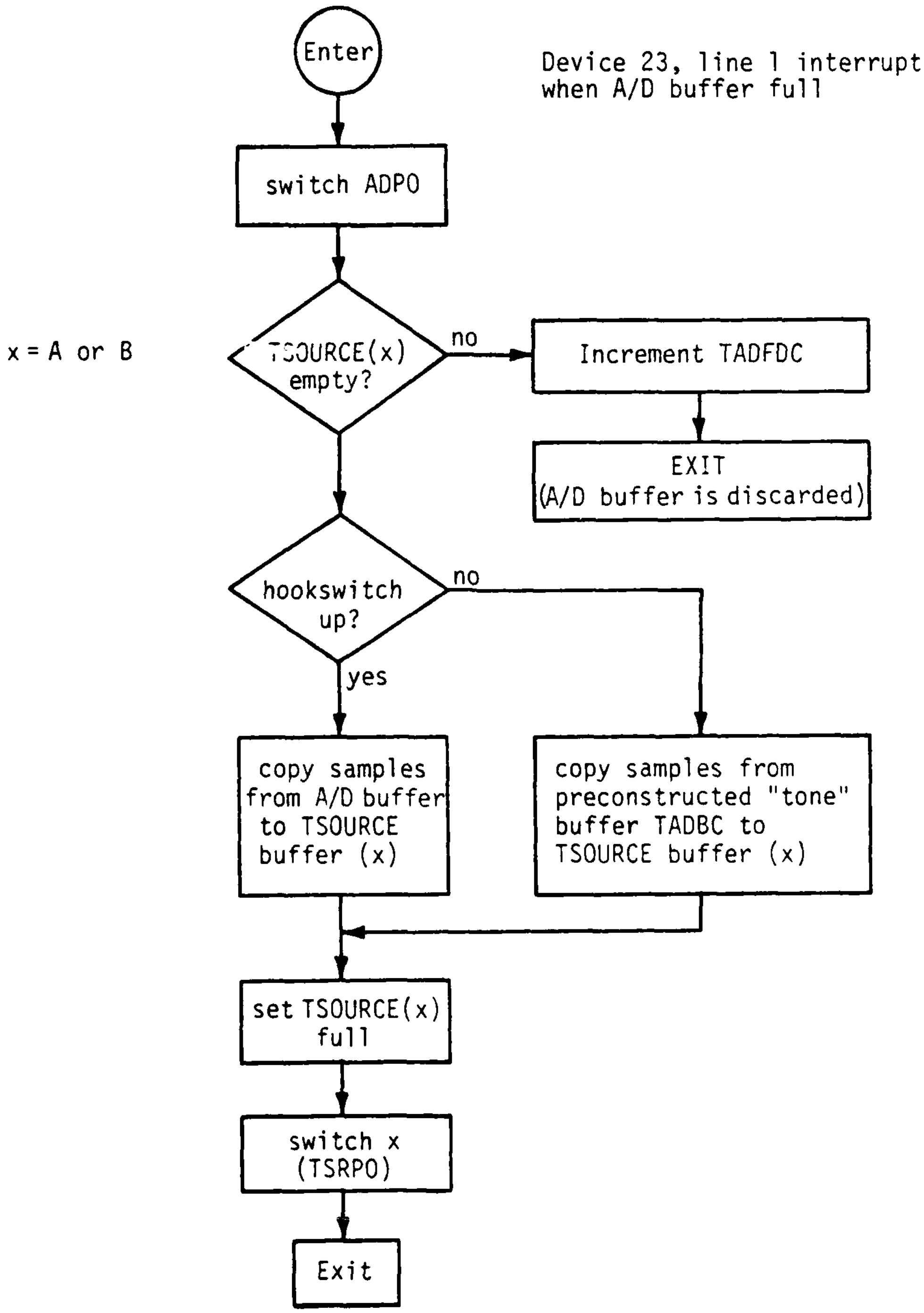


FIG. 6. **A/D INTERRUPT SERVICE ROUTINE**

to nonzero to indicate full, TSRPO is switched, and the routine exits. If, on the other hand, the TSOURCE flag shows that the current TSOURCE buffer is not empty and therefore unavailable, ADAMINT simply exits, effectively discarding the new A/D data. An "A/D Frame Discard Counter" (TADFDC) is incremented by one to keep track of the fact that an A/D buffer was discarded.

2.2.1.3 ANALYZE Module

The ANALYZE Module implements the APC analysis algorithm described in Section 2 of the "Specification of the Optimized 16 kb/s APC Algorithm" (henceforth known as the "Algorithm Specification") (Volume I, Appendix A of this report). As described in Section 2.1.3, two versions of the ANALYZE Module exist, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (ANLZA and ANLZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

This section describes, on a function by function basis, the ANLZA/ANLZB function lists, which appear in subroutine BBN16F of the speech coder software (Appendix D). The specification of alternative parameters for the two function lists is indicated in the descriptions below by a '/' (e.g., 'TSRA/B' indicates TSRA for the ANLZA function list and TSRB for the ANLZB function

list). Each function is either supplied by CSPI as part of the SNAP-II Release 3.5 software system or written by BBN. CSPI-supplied functions are described in detail in CSPI documents [2,3,4]. BBN-written functions are described in detail in Appendix A of this report. The buffers and scalars passed as parameters to the functions are defined in subroutine BBN16C of the speech coder software (Appendix D) and are described in detail in Appendices B and C.

2.2.1.3.1 Begin Function List

MPBFL(ANLZA/B)

This CSPI function specifies the start of the indicated function list.

2.2.1.3.2 Set G-flag

MPGSC(IG3,ISET)

This BBN function causes general-purpose flag G3 to be set, indicating the start of ANLZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement and is not required for proper operation of the speech coding software.

2.2.1.3.3 Monitor Peak Input Level

ADPEAK(TSRA/B)

This BBN function serves to maintain a running maximum (in integer scalar TADPK) of the digitized speech input samples, allowing the user to properly adjust the analog input signal level for maximum A/D range without

clipping. The maximum sample in the current input buffer is compared to the previous maximum, and TADPK is updated if necessary. TADPK is displayed (and reset to zero) during speech coder operation in response to a "T" command.

2.2.1.3.4 Transmitter Frame History Updates

THIST(TSP1,TSP0,TE1,TE11,TRH,TRH0)

This BBN function performs frame history updates on three buffers. The last-frame data from the end of each buffer is moved to the beginning of the buffer for use during current frame processing. (The buffers are referenced by TSP1, TE1, and TRH. The end portions of these buffers are referenced by TSP0, TE11, and TRH0.)

2.2.1.3.5 Preemphasis

DPRE(TSP0,TALPH,TSRA/B,TPHST,TONES)

This CSPI function (Discrete Weighted Pre-emphasis Filter) preemphasizes the input speech (TSRA/TSRB) (as indicated in Section 2.1 of the Algorithm Specification), with filter coefficient TALPH, filter history TPHST, unity weighting vector TONES, and output preemphasized speech TSP0.

2.2.1.3.6 Done with Input

MPWT(PRCSR,AP)
MPIST(TSRFA/B,0)

These CSPI functions clear the appropriate buffer status flag (TSRFA or TSRFB) once the previous function has completed, indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new input speech data).

2.2.1.3.7 Pitch Filter Analysis

The functions in this section perform pitch filter analysis on the preemphasized input speech, as indicated in Section 2.2 of the Algorithm Specification.

SSUM(TDCN,TSP,TFSZI)

This CSPI function sums the extended frame of preemphasized input speech samples (TSP), and multiplies the sum by the negative reciprocal of the extended frame size (TFSZI), producing the negative of the extended frame DC value (TDCN).

VMUL(TWSP,1,TSP,TDCN,THAM0,0)

This CSPI function removes the DC term (TDCN) from the input speech (TSP) and multiplies the result by a Hamming window (THAM0), producing windowed speech in TWSP.

FFTLR(TXSP,1,TWSPZ,TCOST,TXSP)

This BBN function is a slightly modified version of the CSPI function FFTNR (Real to complex FFT, not in place) (described in [4]). The modification involves a reformatting of the complex frequency domain output (TXSP) such that the last real output point appears at the end of the array, rather than in the first imaginary output location (which is known to be zero). The real input (TWSPZ) is a zero-padded copy of the windowed speech in TWSP. TCOST is a pregenerated table of cosine values.

VMSQ(TPSPR,TXSPR,TXSPI)

This CSPI function computes the power spectrum (TPSPR) of the windowed speech by computing the magnitude squared from the DFT real (TXSPR) and imaginary (TXSPI) parts.

VCLR(TPSPI)

This CSPI function clears the imaginary part of the complex buffer containing the real power spectrum. This clear operation must be performed every frame because the following inverse FFT function writes to TPSPI.

FFILR(TRP,1,TPSP,TCOST,TRP)

This BBN function is a slightly modified version of the CSPI function FFINR (Complex to real inverse FFT, not in place) (described in [4]). The modification involves reformatting of the complex frequency domain input (TPSP) corresponding to the modification described above for FFTLR. TPSP is the complex input consisting of real part TPSPR and imaginary part TPSPI. The real output TRP contains the autocorrelation coefficients of the windowed speech.

PITCH(TRPP,TMH,TRP,TIM,T14,TC2P)

This BBN function computes and codes the pitch lag (in terms of number of speech samples) from the autocorrelation coefficients (TRP). The pitch lag is output in TMH, and the coded pitch is output as an integer half-word in the left half of real scalar TIM. -TRP(pitch lag -1), -TRP(pitch lag), and -TRP(pitch lag +1) are output (for the three-tap pitch predictor coefficient computation) in TRPP. The single-tap pitch predictor coefficient is output in real scalar TC2P.

MEWL(TC,TINC,TRPP,TKTHR,TRP,TWORK)

This CSPI function (described in [4]) computes the three-tap pitch predictor coefficients (TC) from the pitch-centered autocorrelation coefficients (TRPP).

STAB(TCH,TC2P,TIC,TC)

This BBN function performs a stability check on the three-tap pitch predictor coefficients (TC) and uses the

single-tap predictor (with zeros for the first and third coefficients, TC2P as the second coefficient) if the three-tap coefficients are found to be unstable. The pitch predictor coefficients are then coded and quantized into buffers TIC and TCH, respectively.

2.2.1.3.8 Inverse Pitch Filter

INVPF(TE10,TMH,TSP1,TCH)

This BBN function inverse filters the preemphasized speech (TSP1) (as indicated in Section 2.3 of the Algorithm Specification), using quantized pitch predictor coefficients (TCH) and pitch lag (TMH) and produces a first residual signal (TE10).

2.2.1.3.9 Spectral Filter Analysis

The functions in this section perform spectral filter analysis on the first residual signal, as indicated in Section 2.4 of the Algorithm Specification.

VMUL(TWE1,1,TE10,0,THAM1,0)

This CSPI function multiplies the first residual signal (TE10) by a Hamming window (THAM1), producing a windowed first residual (TWE1).

DCORZ(TRS,0,TWE1,TWE1)

This CSPI function (described in [4]) computes seven autocorrelation coefficients (TRS) from the windowed first residual (TWE1).

HFC(TRSP,TRS,TLMU)

This BBN function modifies the autocorrelation coefficients (TRS), producing high-frequency-corrected autocorrelation coefficients (TRSP).

MWLQ6(TKE,TKTHR,TKH,TRSP,TIK)

This BBN function is a modified version of the CSPI function MWLD. It uses the Levinson recursion to compute the spectral reflection coefficients from the corrected autocorrelation coefficients (TRSP), and then codes and quantizes them into buffers TIK and TKH.

2.2.1.3.10 Inverse Spectral Filter

The functions in this section perform inverse filtering of the first residual signal, as indicated in Sections 2.4.5 and 2.5 of the Algorithm Specification.

VKTOA(TAH,TKH)

This BBN function converts reflection coefficients (TKH) to spectral predictor coefficients (TAH).

DCORZ(TE2,0,TE1,TAHR)

This CSPI function (described in [4]) inverse filters the first residual (TE1) by performing a correlation operation with the pitch predictor coefficients in reverse order (TAHR) (hence a convolution), producing a second residual signal (TE2).

2.2.1.3.11 Error Protection on Previous Frame Parameter Data

PROPAR(B/A)

This BBN function performs error protection on previous frame parameter data (in buffer TSNKB/TSNKA), depositing error protected and bitstreamed transmission data in buffer TBTA/TBTA. This function executes in the CSPU while the previous function (DCORZ) is executing in the AP. (See Section 2.2.1.3.18 for a more detailed discussion of this module.)

2.2.1.3.12 Noise Shaping Filter Calculation

VMUL(TANS,1,TAH,0,TFAC,0)

This CSPI function computes noise shaping filter coefficients (TANS) as indicated in Section 2.4.6 of the Algorithm Specification.

2.2.1.3.13 Gain Factor Calculation

GAIN(TGFAC,TIG,TIDG,T72R,TE2,T3R)

This BBN function computes and quantizes energy and segment delta-gains as indicated in Section 2.6 of the Algorithm Specification. TIG is the coded energy of the second residual (TE2). TIDG contains the three coded segment delta-gains, and TGFAC contains the three residual quantizer segment scale factors.

The quantization table for energy differs from that in the Algorithm Specification by a scale factor of 2.511*(10**-7) (-66 dB), which accounts for the difference in input speech signal magnitude between the simulation (±1024) and the real-time implementation (±0.5). (The ADAM and AOM have a range of ±1, but that amplitude is rare.) The energy and delta-gain decoding tables differ from their quantization tables by a square root. Quantization is performed on the computed energies, but the APC and synthesis operations use gain (or square root of energy) factors. The gain decoding table also differs from that in the Algorithm Specification by -66 dB (scale factor of 5.01*(10**-4)).

2.2.1.3.14 APC Residual Calculation

APC(TIUA/B,TMH,TSP0,TFILT,TGFAC,TRH,TVH,TQ1)

This BBN function computes and codes the APC residual as indicated in Section 2.7 of the Algorithm Specification. The input preemphasized speech is in TSP0. The spectral, pitch, and noise shaping filter coefficients are contiguous in buffer TFILT. TMH

contains the pitch lag. TRH, TVH, and TQ1 retain frame-to-frame memories of the filters. The coded APC residual is stored in buffer TIUA/TIUB (equivalent to the beginning of TSNKA/TSNKB).

2.2.1.3.15 Error Protection on Previous Frame Residual Data

PRORES(B/A)
MPIST(TBTFB/A,1)

These functions (PRORES is a BBN function; MPIST is a CSPI function) perform bitstreaming on the previous frame's residual data (in buffer TSNKB/TSNKA), depositing bitstreamed transmission data in buffer TBTB/TBTA. The appropriate buffer status flag (TBTFB or TBTFA) is set, indicating that the associated buffer is now full (i.e., ready to be transmitted). These functions execute in the CSPU while the previous function (APC) is executing in the AP. (See Section 2.2.1.3.18 for a more detailed discussion of this module.)

2.2.1.3.16 Transmitter Data Collection

GTHR(TSNKA/B,TIG,TIDG,TIM,TIC,TIK)

This BBN function collects all coded parameter transmission data into a single buffer (TSNKA/TSNKB). (The coded residual data has been placed directly into TSNKA/TSNKB by the APC function).

2.2.1.3.17 End of Function List

MPEFL(ANLZA/B)

This CSPI function specifies the end of the indicated function list.

2.2.1.3.18 Error Protection and Bitstreaming

The PROPAR and PRORES modules take as input a TSINK buffer

(either TSNKA or TSNKB) containing quantized and coded analysis parameters and residual samples and produce as output a TBITS buffer (again, TBTA or TBTB according to the TSINK buffer designation) in which:

1. certain high-order data bits have been grouped together and protected with (7,4) Hamming codewords;

2. the data to be transmitted has been "bitstreamed", one bit per half-word, in the form used by the TMODEM scroll program;

3. histogram information of the coded analysis parameters has been recorded.

These operations are performed in the CSPU, and therefore they can be executed concurrently with an AP operation. These operations are implemented as two separate CSPU modules so that each module's execution time is (mostly) hidden by that of an AP module. The PROPAR module performs the error-protection and bitstreaming of the analysis parameters (gain, delta-gains, pitch, pitch filter coefficients, and reflection coefficients), and it also records the histogram information for those parameters (except pitch). PRORES bitstreams the residual values only.

The format of the TSINK buffer is shown in Table 1, along with the number of bits per parameter and the number of high-order bits protected by Hamming codes. (C1-C3 denote the three pitch filter coefficients, and K1-K6 denote the six reflection coefficients.) A total of 44 high-order bits of the transmitted parameters are protected using 11 Hamming (7,4) codewords.

Word	Parameter	No. of bits	Bits protected
0-215	Residual(216 samples)	2	0
216	Gain	6	6
217-219	Delta-gain (3)	2	2
220	Pitch	7	7
221	C1	3	2
222	C2	4	3
223	C3	3	2
224	K1	6	5
225	K2	5	4
226	K3	4	3
227-229	K4,K5,K6	4	2

TABLE 1. TSINK BUFFER FORMAT

The TBITS buffer contains one data bit in the rightmost bit of each 16-bit half-word. The format of the TBITS buffer (also the frame of data transmitted) is shown in Table 2. (Bits of coded parameters are numbered starting with bit 0 on the right. B5432 thus denotes a 4-bit field of bits 5 through 2.)

A histogram-gathering function was included in the PROPAR module for gathering statistics on the effectiveness of the quantization tables for the analysis parameters and for verifying correct transmitter operation. A PDP-11 program (HIST16) was implemented (not delivered, but shown in Appendix D) to read the histogram buffers defined on Bus 1 and list them in a text file.

2.2.1.4 TMODEM Interrupt Service (TMODEMINT)

The TMODEM Interrupt Service routine is shown in Fig. 7. When activated by a Line 1 interrupt from the modem IOS-2 scroll, this routine updates the TMODEM pointer offset to point to the

Bit No.	Function
0	Sync bit
1-2	1st residual sample (LSB first)
.	.
.	.
431-432	216th residual sample
433-439	B5432 gain (Hamming (7,4) codeword)
440-446	B10 gain, B10 delta-gain-1 (7,4)
447-453	B10 delta-gain-2, B10 delta-gain-3 (7,4)
454-460	B6543 pitch (7,4)
461	B0 C1
462-468	B2 C1, B210 pitch (7,4)
469	B0 C2
470-476	B321 C2, B1 C1 (7,4)
477	B0 C3
478	B0 K1
479-485	B54 K1, B21 C3 (7,4)
486	B0 K2
487-493	B4321 K2 (7,4)
494	B0 K3
495-501	B3 K3, B321 K1 (7,4)
502-508	B32 K4, B21 K3 (7,4)
509-510	B10 K4
511-512	B10 K5
513-514	B10 K6
515-521	B32 K6, B32 K5 (7,4)

TABLE 2. TBITS BUFFER FORMAT

current (just emptied) TMODEM buffer, then checks the current TBITS buffer flag to see if there is new bitstream data ready to be transmitted. If the flag is set to full, the data is copied from the TBITS buffer to the current TMODEM buffer, the TBITS flag is set to empty, and the TBITS pointer offset is updated to point to the other TBITS buffer/flag. If, on the other hand, the current TBITS buffer flag indicates not-full, bitstream data corresponding to a frame of silence is copied (from buffer TBTC) to the current TMODEM buffer, the "fake frame" is counted by

incrementing TMFFC, and the routine exits without having changed the TBITS pointer offset.

In either case, only 521 words (bits) of data are copied to the TMODEM buffer. The data bit of the first word of each TMODEM buffer is not copied into, as it is the synchronization bit: a 0 in TMODEMA and a 1 in TMODEMB. Since data is transmitted alternately from TMODEMA and TMODEMB, the data frames at the receiver have the sync bit alternating between 0 and 1 in successive frames.

2.2.1.5 TMODEM and RMODEM I/O Scroll Program (RTPROG)

The digital data input and output is performed by an IOS-2SM Scroll processor that has been augmented by a modem interface. This interface also contains the dual-rate clock for timing both the modem data and the speech sample input and output.

The TMODEM and RMODEM Scroll program (RTPROG) is shown in Fig. 8. Although the TMODEM output and RMODEM input processes are logically distinct, they are performed by this single program. After setting the rates of the modem-data and speech-sample clocks and initializing buffer switches and address pointers, the program enters a basic loop that checks the two peripheral flags for data ready from the modem receiver interface (P1 set) or for the modem transmitter interface ready to accept data (P2 set).

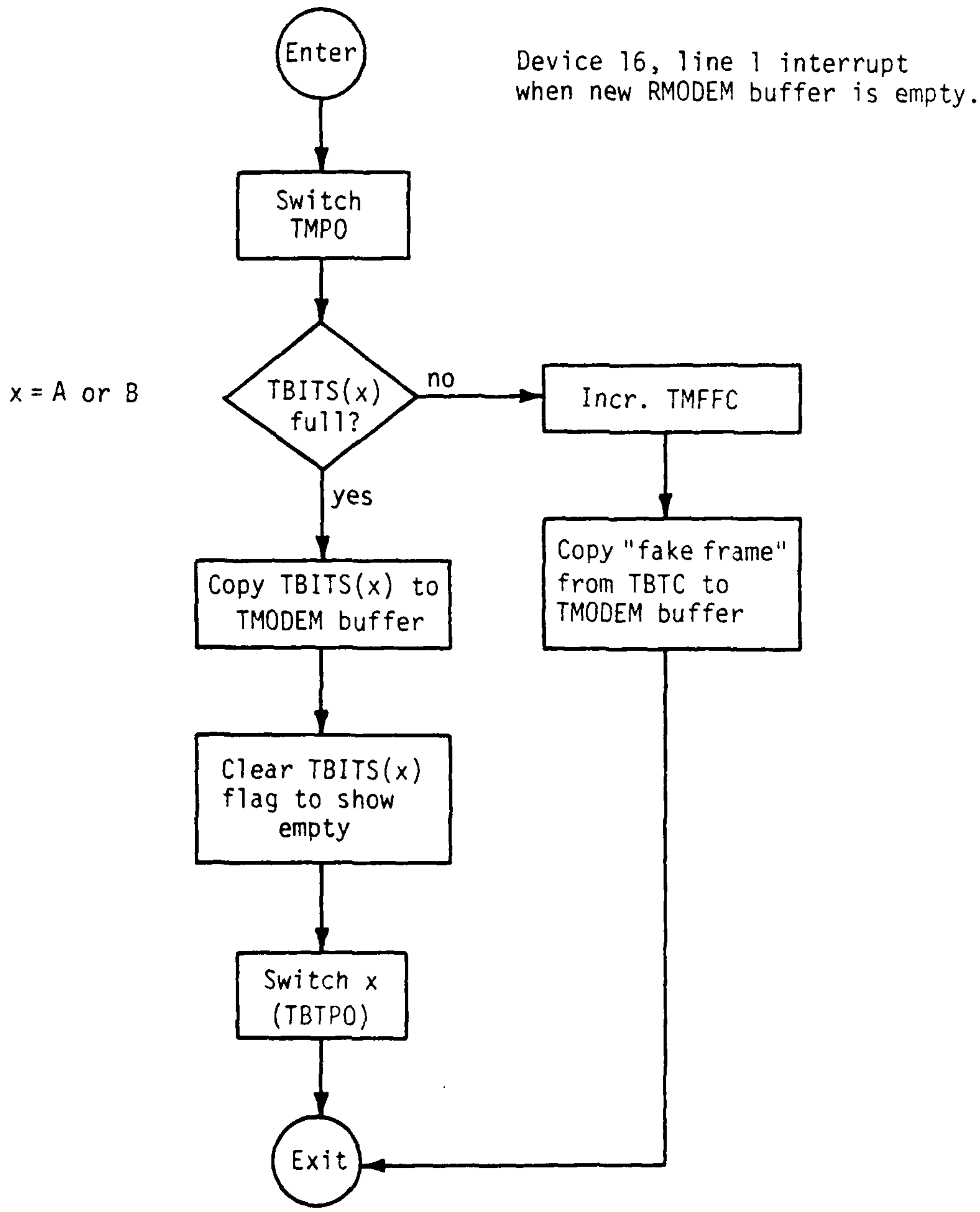


FIG. 7. TMODEM INTERRUPT SERVICE ROUTINE (TMODEMINT)

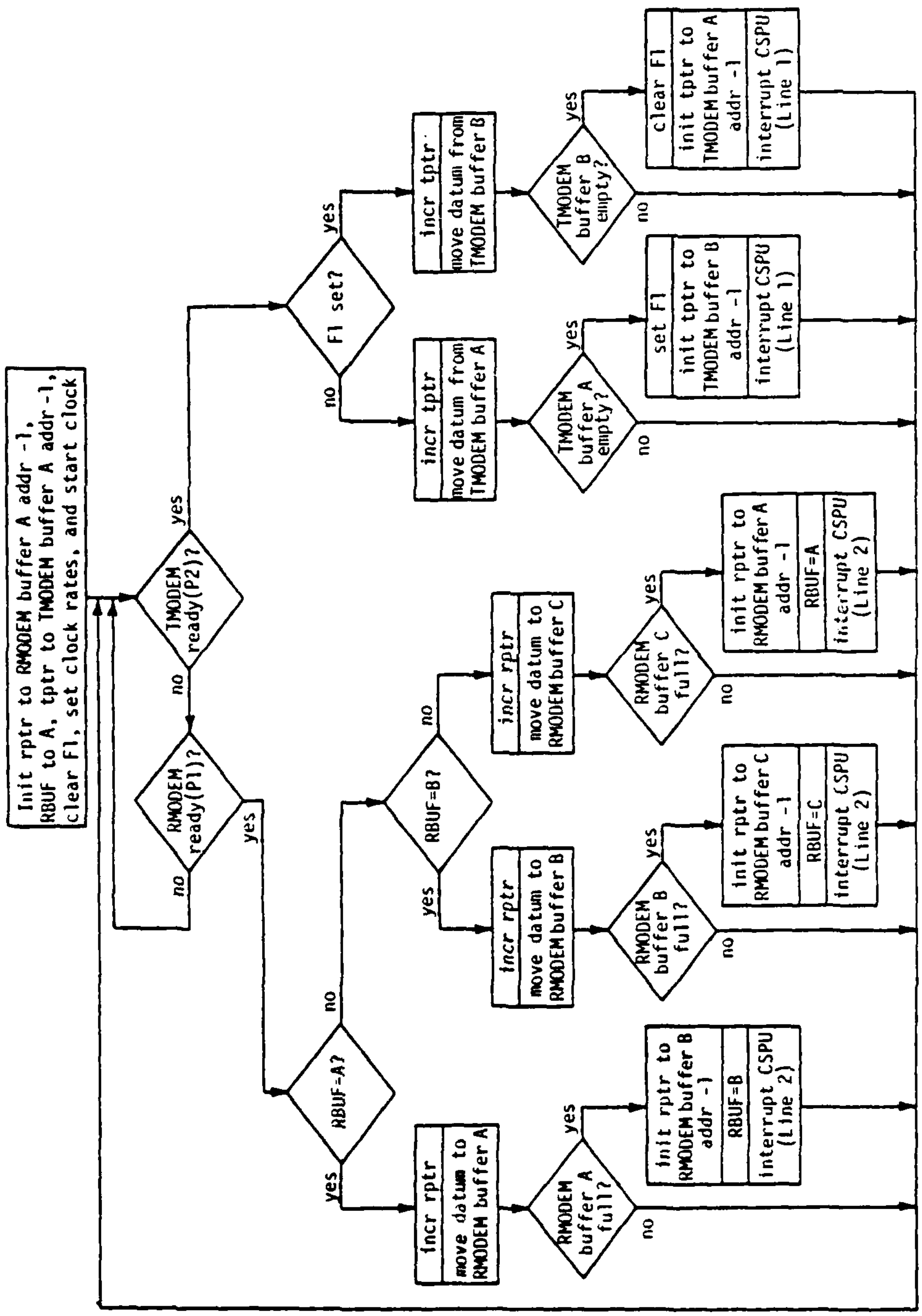


FIG. 8. TMODEM AND RMODEM SCROLL PROGRAM FLOW CHART

When a new datum is available from the interface, it is transferred to RMODEM buffer A, B, or C, depending on the value of a buffer switch in register R0 and an address pointer in register R1. The transfer clears flag P1. If the input buffer is now full, the buffer switch and address pointer are changed to point to the next buffer and the CSPU is interrupted on line 2.

When the interface becomes ready to accept a datum (to be transmitted), a word is transferred from TMODEM buffer A or B, depending on the value of flag F1, used as a buffer switch, and an address pointer in register R2. The transfer clears flag P2. If the output buffer is now empty, the buffer switch and address pointer are changed to indicate the other TMODEM buffer and the CSPU is interrupted on line 1.

2.2.1.6 Transmitter Input/Output Buffer Initial Sequence

The several layers of shared input/output buffers in the speech coder transmitter and receiver require that the modules that reference them be initialized to do so in the proper order. The start-up sequence for the transmitter is described below.

The ADAM scroll program starts by writing speech samples into buffer TADBA (and then proceeds to write into TADBB, etc.). At the first ADAM interrupt, ADAMINT copies from TADBA to TSRA (TSOURCE A buffer) and sets the flag TSRFA. The background process executes ANLZA first, since TSRFA=1 and TBTFB is initially 0. The execution of ANLZA includes PROPAR(B) and

PRORES(B), which write the first bitstream data into buffer TBTB; then the flag TBTFB is set to 1. The TMODEMINT interrupt routine is initialized to expect its first bitstream data in buffer TBTB, so that is the first data copied into a TMODEM buffer for output by the TMODEM scroll program.

Meanwhile, the TMODEM scroll program has started by transmitting (initialized) data from buffer TMDMA. It then sends an interrupt and proceeds to read from TMDMB, and so forth. The initial TMODEM interrupt activates TMODEMINT, whose first task is to provide new data for the just-emptied TMDMA buffer. TMODEMINT is initialized to check the flag TBTFB, as described above. At the end of the first frame, however, ANLZA has not yet executed and there is as yet no data in TBTB, so TMODEMINT copies "dummy" bitstream data instead from TBTC into TMDMA and exits. The next time TMODEMINT is activated, it again checks TBTFB and (since ANLZA has run by now) finds it set, so it copies the new bitstream data from TBTB to the (just-emptied) TMDMB buffer. Thereafter all data buffers are alternately filled and emptied without conflict.

2.2.2 Receiver

The Receiver is similar in structure to the Transmitter. The Receiver is shown in Fig. 9.

Data is accepted from the modem and put into one of three

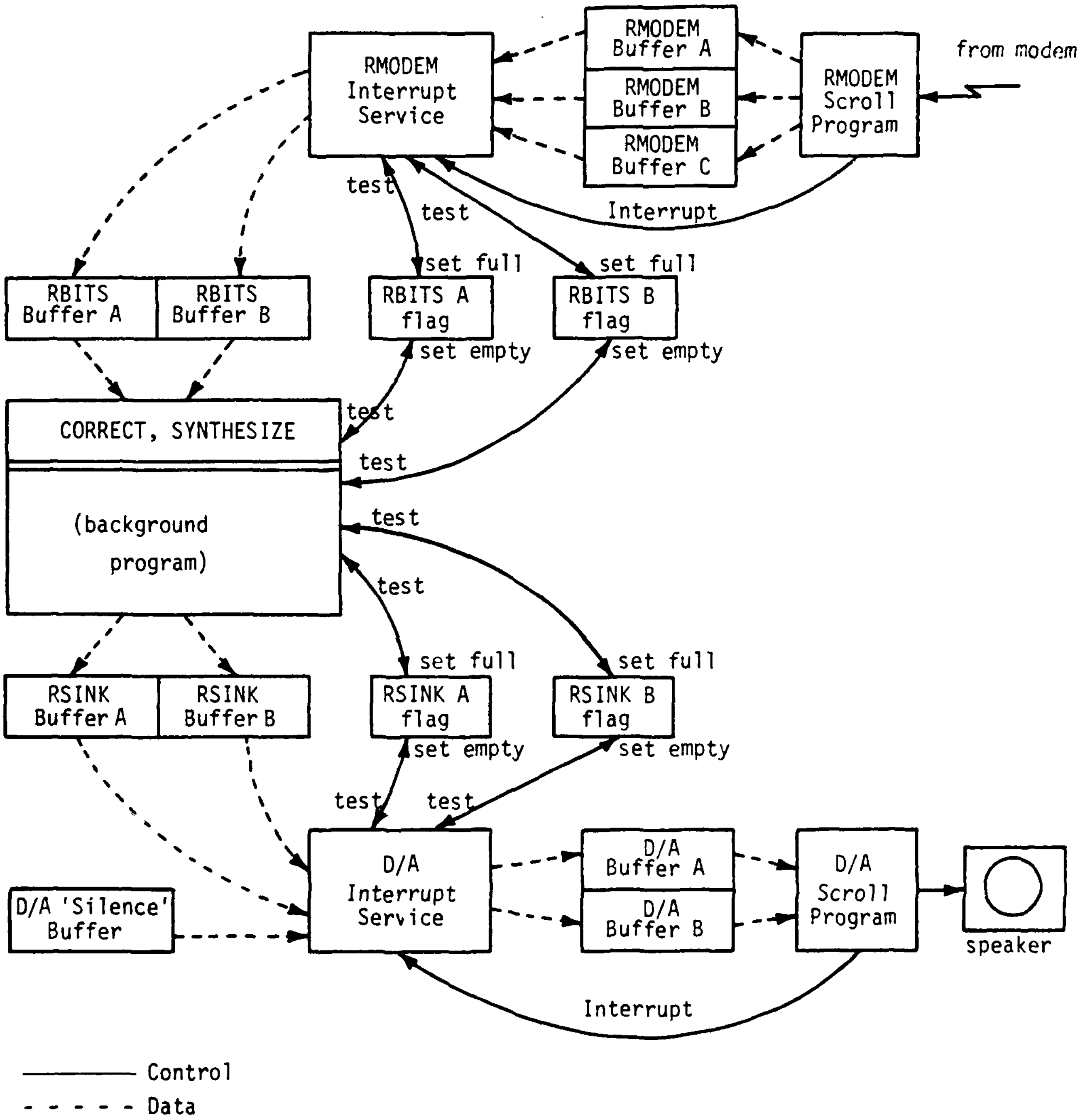


FIG. 9. RECEIVER PROCESS

RMODEM buffers by the RMODEM program running in an I/O scroll. When this program fills a buffer, it generates an interrupt to the CSPU and begins filling the next buffer (A,B,C,... in rotation).

This interrupt activates the RMODEM Interrupt Service routine, which checks frame synchronization and, if correct, transfers the new data to an empty RBITS buffer and sets the corresponding RBITS flag to indicate full. The data that is actually transferred does not correspond exactly to the just-filled RMODEM buffer. Rather, it is a "frame" of data completely contained in the new RMODEM buffer and the RMODEM buffer previously filled. A frame of data begins with a sync bit. Since this bit may occur anywhere within a single RMODEM buffer, two full buffers (and therefore three buffers altogether) are required to guarantee that a complete frame is available.

The CORPAR and DECODA/B modules, running in the CSPU at background level, empty the full RBITS buffer, perform error correction and decoding of the incoming bitstream, copy these decoded parameters into an (empty) RSOURCE buffer, and clear the RBITS flag to signify empty.

The SYNTHESIZE module, running in the CSPU at background level, empties the (full) RSOURCE buffer, performs the processing necessary to synthesize speech, and puts the output speech samples into an (empty) RSINK buffer, setting the corresponding

RSINK flag to indicate full. The SYNTHESIZE module, although logically following the CORPAR and DECODA/B modules, in fact contains them and is executed concurrently with them. That is, SYNTHESIZE operates on the data supplied by the previous execution of CORPAR and DECODA/B. The first time that SYNTHESIZE is executed, it operates on a RSOURCE buffer that has not been filled by CORPAR and DECODA/B; therefore the RSOURCE buffers are initialized to contain data that will synthesize silence.

The data from the RSINK buffer is transferred to an empty D/A buffer by the D/A Interrupt Service routine. This routine also sets the corresponding RSINK flag to indicate empty.

This Interrupt Service routine is activated by an interrupt from the D/A Scroll program, running in another I/O scroll (the AOM, or Analog Output Module). This scroll program transfers data from a D/A buffer to the D/A converter, interrupts the CSPU when the buffer is empty, and begins transferring data from the other D/A buffer.

2.2.2.1 RMODEM Scroll Program

The RMODEM scroll program is described in Section 2.2.1.5.

2.2.2.2 RMODEM Interrupt Service (RMODEMINT)

The RMODEM Interrupt Service routine is shown in Fig. 10. It is activated by a line 2 interrupt from the modem IOS-2 scroll, which indicates that the next RMODEM buffer is full. The

routine updates the RMODEM pointer offset to point to the just-filled buffer. Then it copies the on-hook bit from the first word in the current RMODEM buffer to the integer scalar RONHK, where it is saved for use by the ADAM interrupt service routine (Section 2.2.1.2).

Then the integer scalar RERRS is tested. This is normally zero, but if it is set to nonzero (by a command from the host program), a channel error simulation routine (RMI$SIM) is called. RMI$SIM uses the value in RERRS as the number of errors to be introduced (quasi-randomly) per frame. Since there are 522 bits per frame, each error contributes about 0.2% to the error rate. This error simulation is intended only for demonstration; it is not an accurate simulation of an errorful channel.

The next set of operations deals with frame synchronization. Since the receiver has no prior knowledge about the position of the frame boundaries in the input data stream, it must infer the boundary position from the received data pattern. The first bit of each transmitted frame is the sync bit, which alternates between 0 and 1 in successive frames; this pattern allows the receiver to detect and maintain frame synchronization. The synchronization routines are described in Section 2.2.2.2.1 below.

RMODEMINT tests the value of RSYNC, an integer scalar that maintains the receiver's sync-state. If RSYNC=0 (as it is when

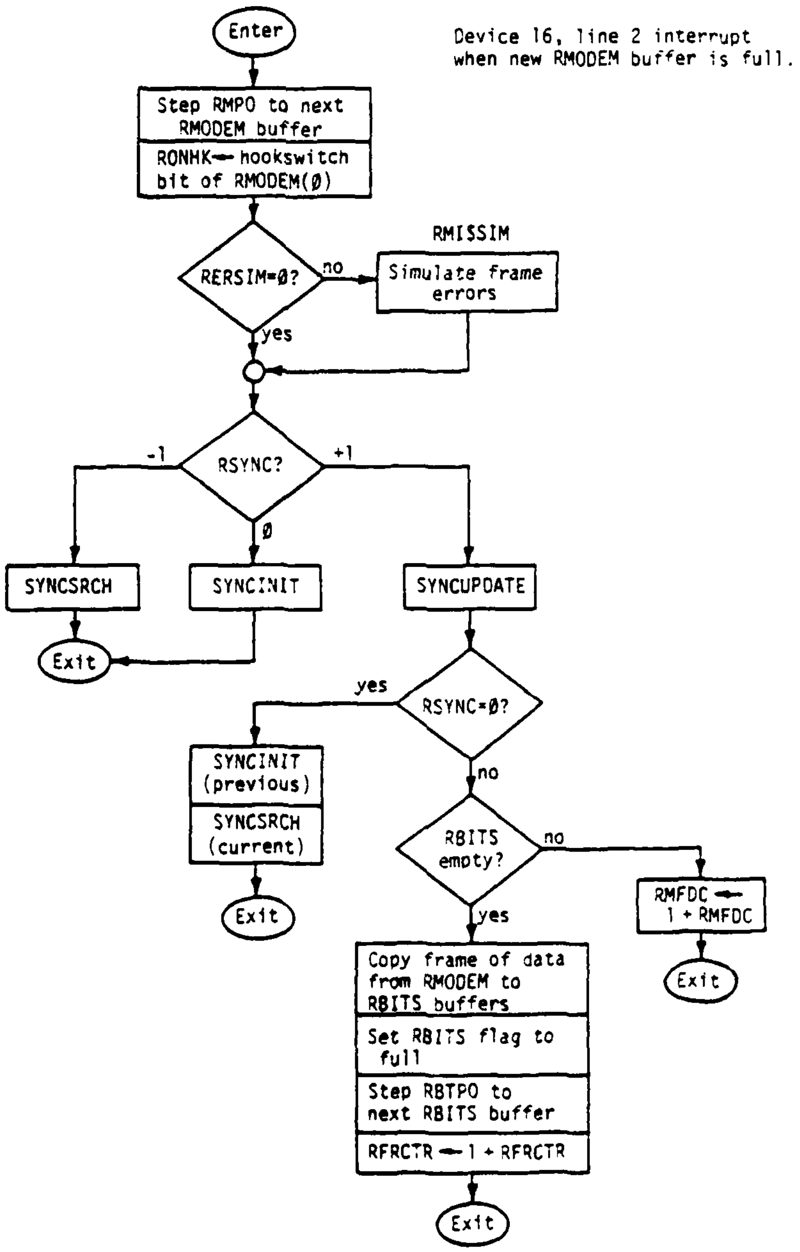


FIG. 10. RMODEM INTERRUPT SERVICE MODULE (RMODEMINT)

the speech coder is initialized), nothing is known about sync, so the SYNCINIT routine is called to initialize the sync-search process with the current RMODEM buffer. If RSYNC<0, then the receiver is still searching for sync, so the SYNCSRCH routine is called to continue the search using the new data in the current RMODEM buffer. In both of these cases, RMODEMINT exits immediately. If RSYNC>0, then the receiver has gained frame-sync, so the SYNCUPDATE routine is called to check that the current RMODEM buffer continues to show the expected sync-bit value. Upon the return from SYNCUPDATE, RSYNC is checked again; if it has been set to zero, then the receiver has lost sync and must again search the incoming bitstream for the sync-bit pattern; therefore SYNCINIT and SYNCSRCH are called to restart sync-searching with the previous and current RMODEM buffers.

If RSYNC is still greater than zero after SYNCUPDATE, then synchronization is confirmed, and the receiver makes use of the data for speech output. RMODEMINT checks the current RBITS buffer flag; if empty, the most recent frame of data is copied from the RMODEM buffers to the current RBITS flag, the RBITS flag is set to full, and the RBITS pointer offset (RBTPO) is switched to the next RBITS buffer. Note that in general, the current <u>frame</u> will straddle the previous and current RMODEM buffers. It is for this reason that there are <u>three</u> RMODEM buffers, two to hold the current frame while the third is being filled by the RMODEM scroll program.

If, for some reason, the RBITS buffer is not empty (ready to accept new data), the new data frame is effectively discarded. A frame discard counter (RMFDC) is incremented to take note of this, and the routine exits without updating RBTPO.

2.2.2.2.1 Synchronization Routines

The SYNCINIT subroutine (Fig. 11) initializes the state of the frame synchronization section of the speech coder.

Its primary function is to copy the data bit of each word of the current RMODEM buffer into RSSPF, the "previous frame" memory used in SYNCSRCH, and to clear the counts in the RSSSS buffer. SYNCINIT also zeros the gain word of both RSOURCE buffers and the entire previous RBITS buffer, so that when the speech coder receiver resumes operation after regaining sync, the incorrect information in these buffers will not produce unwanted transients in the speech output.

The SYNCSRCH subroutine (Fig. 12) scans the incoming RMODEM buffers (one per call), building up statistics on the frame-to-frame data patterns in each bit position of the frame until it detects a large enough number (ACQTHR) of consecutive bit alternations in a single bit position to allow it to identify that position as carrying the sync bit.

When that occurs, it sets the value of RBOFO ("beginning-of-frame-offset") to the distance from the start of the buffer to

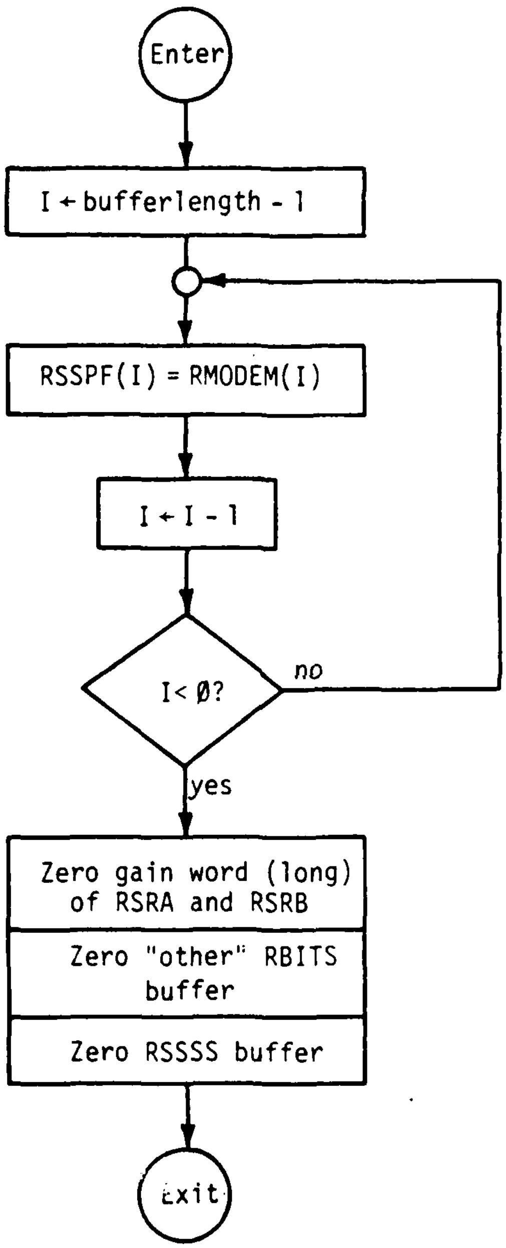


FIG. 11. FRAME SYNCHRONIZATION INITIALIZATION MODULE (SYNCINIT)

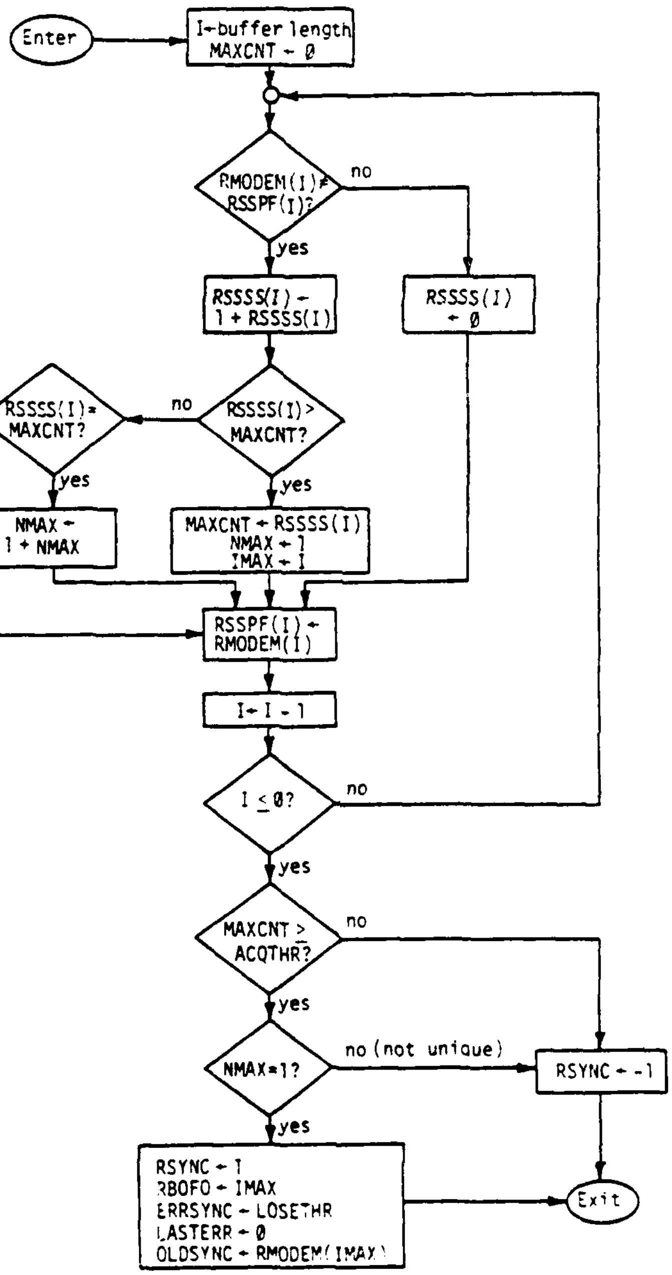


FIG. 12. FRAME SYNCHRONIZATION SEARCH MODULE (SYNCSRCH)

the sync-bit position and RSYNC to +1 to cause RMODEMINT to start passing input frames to the speech coder synthesizer on the next RMODEM interrupt. SYNCSRCH also initializes three variables that are used in subsequent calls to SYNCUPDATE.

The SYNCUPDATE subroutine (Fig. 13) is used once sync has been acquired, to check the sync-bit position of each incoming RMODEM buffer to see if it has the expected value.

It is the function of SYNCUPDATE to declare sync lost if the declared sync bit does not show the expected frame-to-frame alternation. Since channel errors may cause the sync bit to be incorrect, the loss-of-sync criterion must be more stringent than a single incorrect sync bit. The criterion is a sufficient number (LOSETHR) of incorrect sync bits without two consecutive correct sync bits. A simpler criterion, such as LOSETHR consecutive sync bit errors, would be unsatisfactory, since the high-order bit of many speech coder parameters (e.g., energy) may have the same value for many consecutive frames, and a constant bit would compare "correctly" with an alternating bit sequence every other frame.

The performance of the synchronization routines is controlled by two thresholds, ACQTHR and LOSETHR. It is necessary to set ACQTHR high enough so that it is very unlikely that SYNCSRCH will detect sync on the wrong bit position. The value of LOSETHR must be high enough that SYNCUPDATE is unlikely

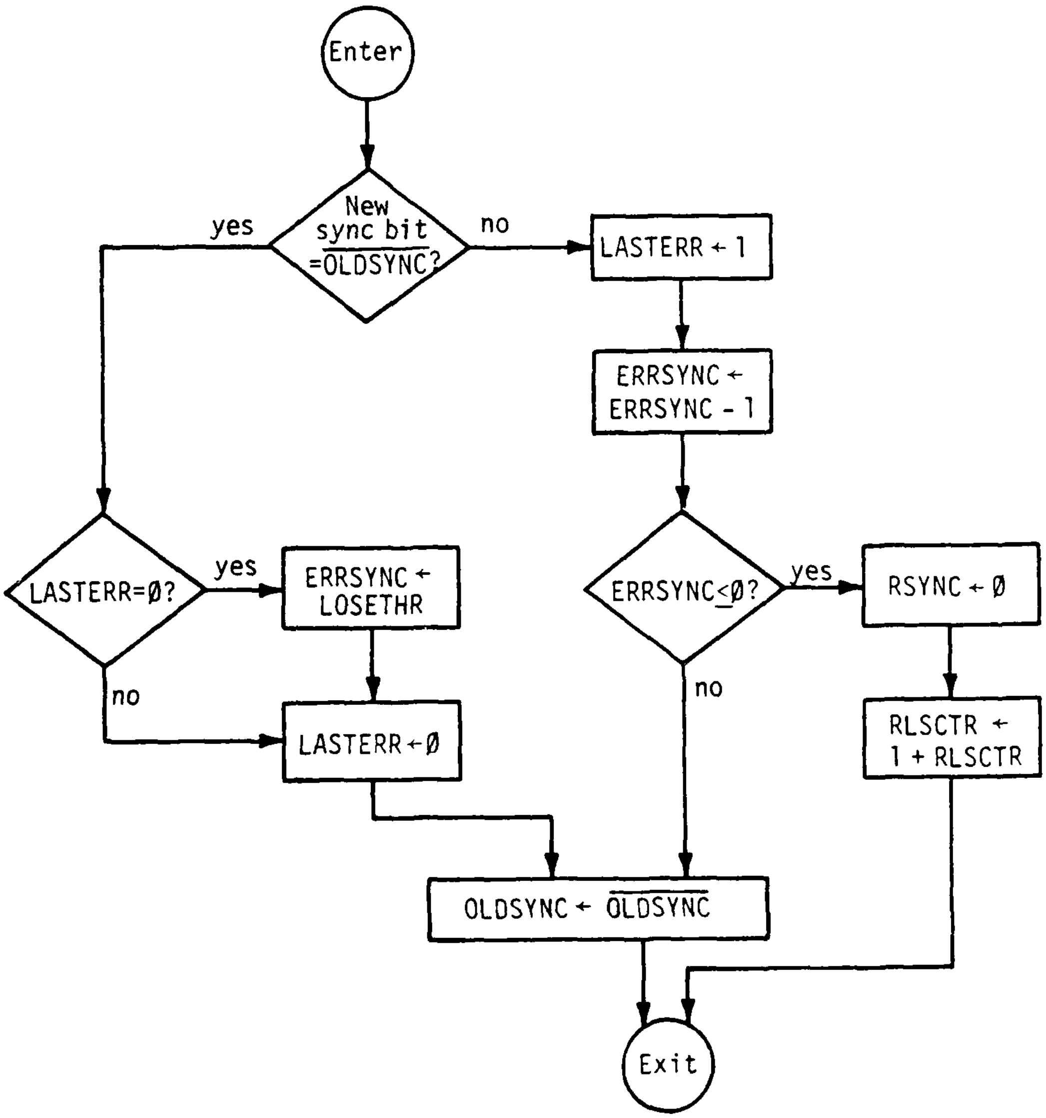


FIG. 13. FRAME SYNCHRONIZATION UPDATE MODULE (SYNCUPDATE)

to declare loss-of-sync due to channel errors falling on the sync bit, but small enough that if sync should be lost for some reason, the receiver does not synthesize too much "garbage" before detecting the loss-of-sync. Of course, while the receiver is searching for sync, RMODEMINT does not pass data to the synthesizer, so silence is output. For a 1% channel bit error rate, the values of ACQTHR=10 and LOSETHR=4 give a probability of acquiring false sync of about 0.0002 and an expected time to spontaneous loss-of-sync of about 10 hours, respectively.

2.2.2.3 SYNTHESIZE Module

The SYNTHESIZE Module implements the APC synthesis algorithm described in Section 3 of the Algorithm Specification. As described in Section 2.1.3, two versions of the SYNTHESIZE Module exist, one for each of the two sets of input/output buffers required for double buffering. Each version consists of a function list of MAP-300 function calls. The two versions (SYNZA and SYNZB) are identical in terms of the sequence of functions called and differ only in the parameters (buffers and/or scalars) passed to several of the functions.

As in the case of Section 2.2.1.3, this section describes, on a function by function basis, the SYNZA/SYNZB function lists, which also appear in subroutine BBN16F.

2.2.2.3.1 Unbitstreaming and Error Correction

The CORPAR and DECODA/B modules take as input an RBITS

buffer containing a frame of bitstream data input from the modem and produce as output an RSOURCE buffer containing error-corrected and decoded floating point values of analysis/synthesis parameters and residual samples, ready for immediate use by the speech coder synthesizer.

The format of the RBITS buffer is the same as the TBITS buffer, shown in Table 2 above. The data bits are regrouped, and the 7-bit codewords are error-corrected by lookup in an inverse Hamming code table. The bits are then grouped into coded parameter values and decoded by lookup in parameter-specific decoding tables of floating-point values. (As noted in Section 2.2.1.3.13, the gain and delta-gain decoding tables differ from the ones in the Algorithm Specification by a scale factor.) The format of the RSOURCE buffer is similar to that of the TSINK buffer (Table 1), except that the half-word coded TSINK values are replaced by full-word (32-bit) floating point values.

As in the case of the error-protection and bitstreaming operation, the unbitstreaming, error-correction, and decoding operation is performed by two separate modules, which are part of the SYNTHESIZE function list. CORPAR does the operations on the analysis parameters, while DECODA/B performs unbitstreaming and decoding on the residual samples. Unlike the other "bit-pushing" modules, DECODA/B is an AP module. (There are two distinct DECOD modules. DECODA decodes from buffer RBTA to buffer RSPA; DECODB

decodes using the other pair of buffers. The SNAP-II Executive permits the definition of only 63 distinct data buffers; this proved to be too few to allow the definition (to SNAP-II) of the RBITS buffers. Therefore, two DECOD modules for the APS were created, each with the parameters of its data buffers "hard bound" in the source code.)

2.2.2.3.2 Begin Function List

MPBFL(SYNZA/B)

This CSPI function specifies the start of the indicated function list.

2.2.2.3.3 Clear G-flag

MPGSC(IG3,ICLR)

This BBN function causes general purpose flag G3 to be cleared, indicating the start of SYNZ processing. This use of flag G3 is intended for system debugging, timing, and internal measurement and is not required for proper operation of the speech coding software.

2.2.2.3.4 Receiver Data Distribution

DEAL(RSRA/B,RGH,RDGH,RMH,RCH,RKH)

This BBN function distributes all decoded parameter data from a single buffer (RSRA/RSRB) to separate buffers and scalars. (The decoded residual data will be read directly from RSRA/RSRB by the SCLRES function.)

2.2.2.3.5 Receiver Frame History Buffer Update

VMOV(RRH,RRH0)

This CSPI function performs a frame history update on a single buffer. The last-frame data from the end of the buffer (RRH0) is moved to the beginning of the buffer (RRH) for use during current frame processing.

2.2.2.3.6 Residual Scaling

SCLRES(RWH,RGH,RDGH,RUHA/B)

This BBN function computes quantizer scale factors and scales the received APC residual (RUHA/RUHB are equivalent to the beginning of the RSRA/RSRB) into buffer RWH, as indicated in Sections 3.2 and 3.3 of the Algorithm Specification.

2.2.2.3.7 Spectral Synthesis

VLTSY(RVH,RVHM,RKH,RWH)

This BBN function performs a lattice-form spectral synthesis filtering operation on the scaled APC residual (RWH) as indicated in Section 3.5 of the Algorithm Specification, using reflection coefficients (RKH) directly. The output is in RVH. (This function implements an 8-pole filter; we are using it with the last two coefficients zeroed to implement a 6-pole filter.)

2.2.2.3.8 Decoding of Previous Frame Receiver Residual Data and Error Correction on Previous Frame Receiver Parameter Data

DECODB/A
CORPAR(B/A)
MPWT(PRCSR,AP)
MPIST(RBTFB/A,0)

These functions (DECODB/DECODA and CORPAR are BBN-written; MPWT and MPIST are CSPI-supplied) decode the previous frame received residual samples and error-correct the previous frame received parameter data (from RBTB/RBTA), depositing decoded data in buffer RSRB/RSRA. The appropriate buffer status flag (RBTFB or RBTFA) is cleared, indicating that the associated buffer can now be considered empty (i.e., ready to be filled with new received coded data). The last three function execute in the CSPU while the first executes in the AP. (See Section 2.2.2.3.1 for a more detailed discussion of DECODA/DECODB and CORPAR.)

2.2.2.3.9 Pitch Synthesis

PITSYN(RRH,RMH,RVH,RCH)

This BBN function performs a pitch synthesis filtering operation as indicated in Section 3.6 of the Algorithm Specification. The input is in RVH, and the output is in RRH.

2.2.2.3.10 Deemphasis

DFL22(RRHDA/B,RDCF0,RRH0,RDMY0)

This CSPI function does signal deemphasis as indicated in Section 3.7 of the Algorithm Specification. The filter coefficients are in four consecutive scalars starting with RDCF0. The filter memory is in four consecutive scalars starting with RDMY0. The synthesized speech is stored in RRHDA/RRHDB (equivalent to RSNKA/RSNKB).

2.2.2.3.11 Output Ready

MPWT(PRCSR,AP)
MPIST(RSNFA/B,1)

These CSPI functions set the appropriate buffer status flag (RSNFA or RSNFB) once the previous function has

completed, indicating that the associated buffer is full (i.e., ready to be emptied).

2.2.2.3.12 End of Function List

MPEFL(SYNZA/B)

This CSPI function specifies the end of the indicated function list.

2.2.2.4 D/A Interrupt Service Routine (AOMINT)

AOMINT is activated by each AOM line 1 interrupt, signifying the emptying of a D/A output buffer by the AOM scroll program. Its operation, as illustrated by Fig. 14, is analogous to that of the TMODEMINT routine, the only difference being the copying of synthesis output data from RSINK buffers to D/A buffers for output to the AOM. If an RSINK buffer is not available (full), a buffer of silence is output in its place. This happens whenever the speech coder is not receiving data from the remote speech coder and during frame synchronization or resynchronization.

2.2.2.5 D/A Scroll Program (DAPROG)

Speech output is performed by the AOM, an IOS-2 scroll processor that contains two D/A converters. The AOM gets its sample output clock from the SPI, and it sends the D/A output signal to the SPI for subsequent low-pass filtering and output. The signal is in the range -5 to +5 volts, and the sample rate is 6.621 kHz, the same as for the A/D input.

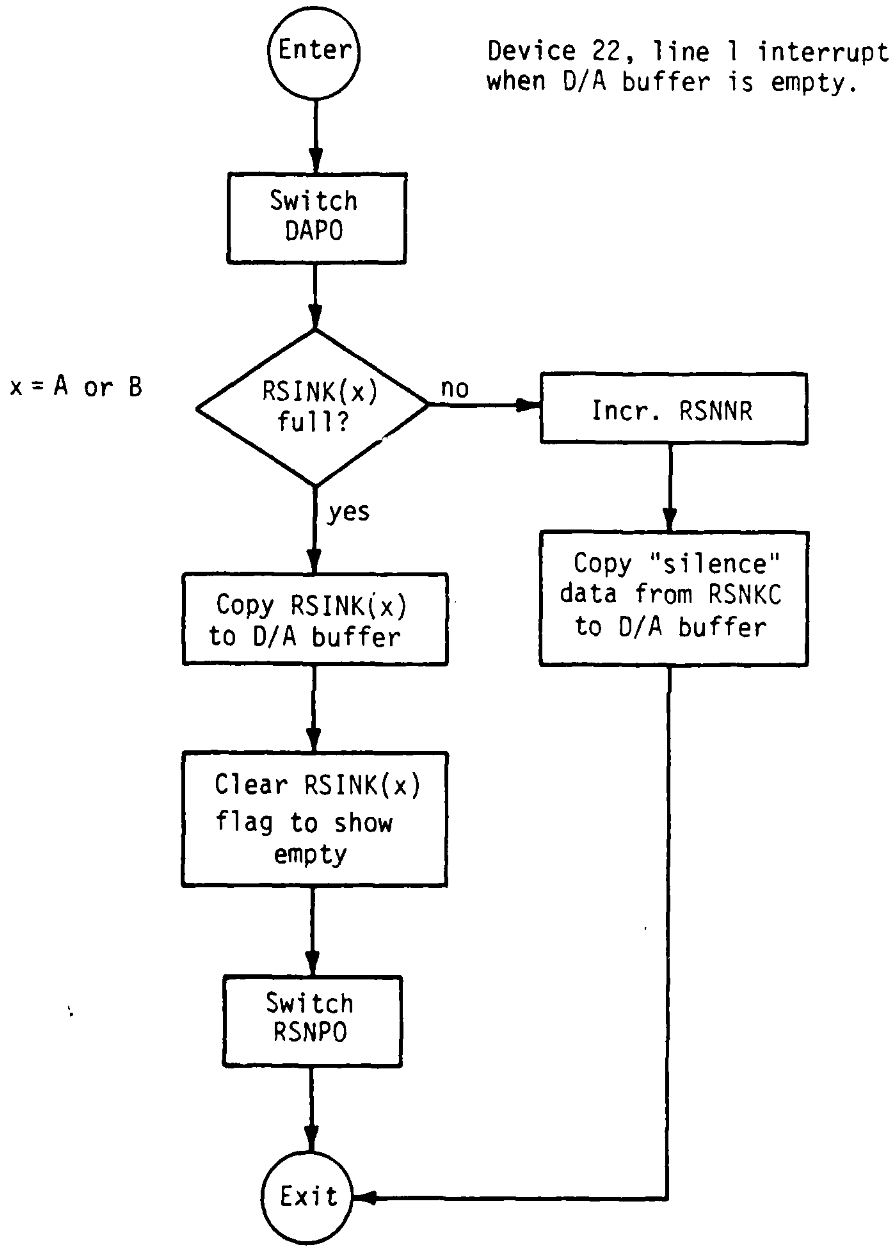


FIG. 14. D/A INTERRUPT SERVICE ROUTINE (AOMINT)

The AOM program (DAPROG) is illustrated in Fig. 15. In "single-channel" mode, the AOM outputs an internally-generated value on D/A Channel 0 while it converts speech samples from MAP memory on D/A Channel 1. The Channel 0 signal is a ramp that is reset each frame time; it is irrelevant to speech coder operation, but it may be used for synchronization or horizontal sweep on an external oscilloscope.

The operation of DAPROG is analogous to that of ADPROG. Output begins from buffer RDABA, and when the end of that buffer is reached, a line 1 interrupt is sent to the CSPU, and buffer RDABB is selected. Reaching the end of RDABB produces another line 1 interrupt and a switch to RDABA, and so forth.

2.2.2.6 Receiver Input/Output Buffer Initial Sequence

The buffer start-up sequence for the speech coder receiver is described below.

The RMODEM scroll program starts by writing received data words into buffer RMDMA, sending an interrupt when it is filled. Upon the initial interrupt, RMODEMINT begins by using RMDMA for frame-sync initialization. Since frame-sync has not yet been acquired, RMODEMINT does not output any data. Successive RMODEM interrupts cause RMODEMINT to use buffers RMDMB, RMDMC, RMDMA, etc. for further frame-sync searching until, after at least 12 input frames, sync has been found. Then RMODEMINT will copy a frame of bitstream data from the RMODEM buffers to buffer RBTA

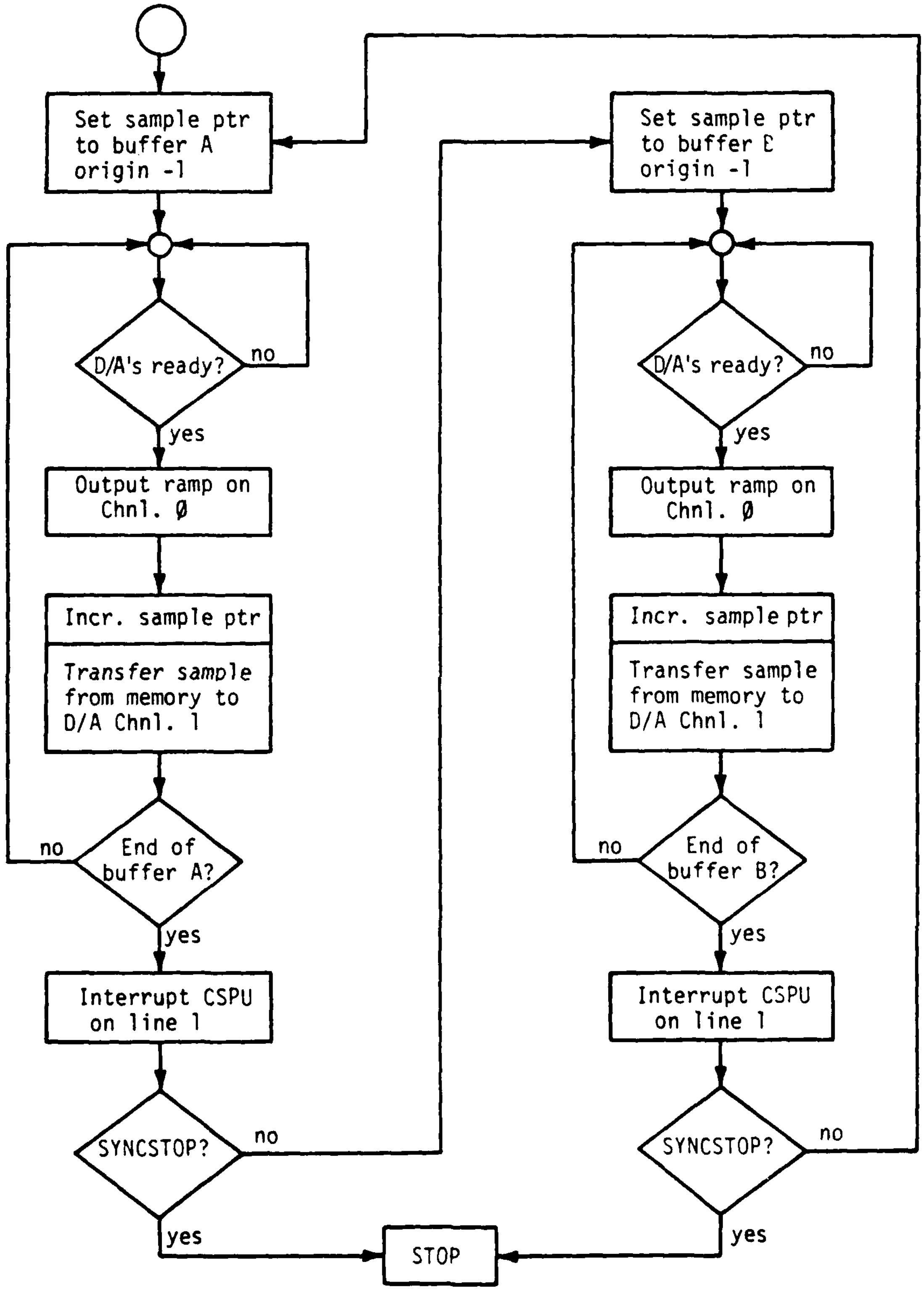


FIG. 15. DAPROG: AOM SCROLL PROGRAM

and set the flag RBTFA=1. The background process will execute SYNZB first, since RBTFA=1 and RSNFB is initialized to 0. SYNZB writes the first synthesized speech data in buffer RSNKB and sets the flag RSNFB=1. The AOMINT interrupt routine is initialized to expect the first synthetic speech data in buffer RSNKB, so when it is next activated, that buffer is copied into a D/A buffer for output by the AOM scroll program.

Meanwhile, the AOM scroll program has started by reading data (initialized to silence) from buffer RDABA, sending an interrupt, going on to RDABB, etc. These AOM interrupts activate AOMINT, whose task is to copy new synthesized speech data from RSINK buffers (initially RSNKB) to the just-emptied D/A buffers (initially RDABA). On its first several (usually 13) activations, AOMINT finds RSNKB empty, because the receiver has not yet gained frame-sync, so AOMINT copies a buffer of silence from buffer RSNKC to the D/A buffers instead. On the next activation after SYNZB has run, AOMINT finds RSNKB full, so it copies it to the D/A buffer, and thereafter all data buffers are alternately filled and emptied without conflict.

2.3 SYSTEM HARDWARE

The 16 kb/s speech coder system is implemented on a MAP-300 array processing computer, which is manufactured by CSP Inc., of Billerica, Mass. Section 2.3.1 describes the configuration of

MAP-300 equipment that is necessary for the speech coder system. Section 2.3.2 describes the additional audio signal interface and IOS-2SM scroll modifications for connection to a modem that complete the system.

2.3.1 MAP-300 Hardware

The MAP-300 configuration that was specified by Defense Communications Agency for the implementation of the speech coder system is listed below.

1	1030	MAP-300 Processor
1	2030	8K x 32 MOS Master Memory, 500 nsec, Bus 1
1	2050	16K x 32 MOS Slave Memory, 500 nsec, Bus 1
1	2203	8K x 32 MOS Master Memory, 300 nsec, Bus 2
1	2410	4K x 32 MOS Master Memory, 170 nsec, Bus 3
1	3110	PDP-11 Interface
1	4020	Model 2SM I/O Scroll
2	4040	Bus Switch (for Model 2SM I/O Scroll)
1	5120	Analog Data Acquisition Module
1	5130	Analog Output Module
1	6100	Expansion Chassis
1	6200	Auxiliary Power Supply

2.3.2 Audio and Modem Interface Hardware

In addition to the MAP-300 equipment listed above, two other pieces of equipment enable the MAP-300 to function as a complete, stand-alone speech coder system: an audio signal interface and a digital data interface to a modem. These two items were designed and built by the GTE Sylvania Electronic Systems Group in Needham, Mass. (see [1], Appendix A). Identical interfaces were provided for all three DCA MAP-300 systems so that they would be interchangeable at the hardware level.

The audio signal interface consists of a handset, tape input and output jacks, and circuitry for the amplification, equalization, and filtering necessary for interfacing speech input and output signals to the MAP-300 A/D and D/A converters.

The modem interface consists of modifications to the MAP-300 IOS-2SM scroll processor for the purpose of transferring data from MAP memory to the modem and also data from the modem into MAP memory. Two real-time clocks derived from a single master oscillator are also provided for controlling the modem data rate and the speech sampling rates.

2.4 SYSTEM SOFTWARE

The speech coder software consists of two distinct sets of modules. The first set is made up of modules that run in the MAP-300, including CSPI-supplied as well as BBN-written programs. Section 2.4.1 describes the MAP-300 modules that must be loaded into the MAP-300 processor before the speech coder can be operated.

The second set of software modules consists of FORTRAN routines that run in the PDP-11. These routines make use of the CSPI-supplied SNAP-II software system to initialize and start the MAP-300 speech coder programs. Section 2.4.2 describes these routines.

2.4.1 MAP-300 Software Components

All MAP-300 processing is done in conjunction with Release 3.5 of the CSPI-supplied SNAP-II software system. For the most part, BBN-written MAP-300 programs take the form of new AP or CSPU functions, callable via the standard SNAP-II calling procedure. In addition, BBN-written interrupt service routines and input/output scroll programs have been added to the executive.

CSPI-supplied MAP-300 software necessary for speech coder operation is described in Section 2.4.1.1. MAP-300 speech coder software written by BBN is described in Section 2.4.1.2.

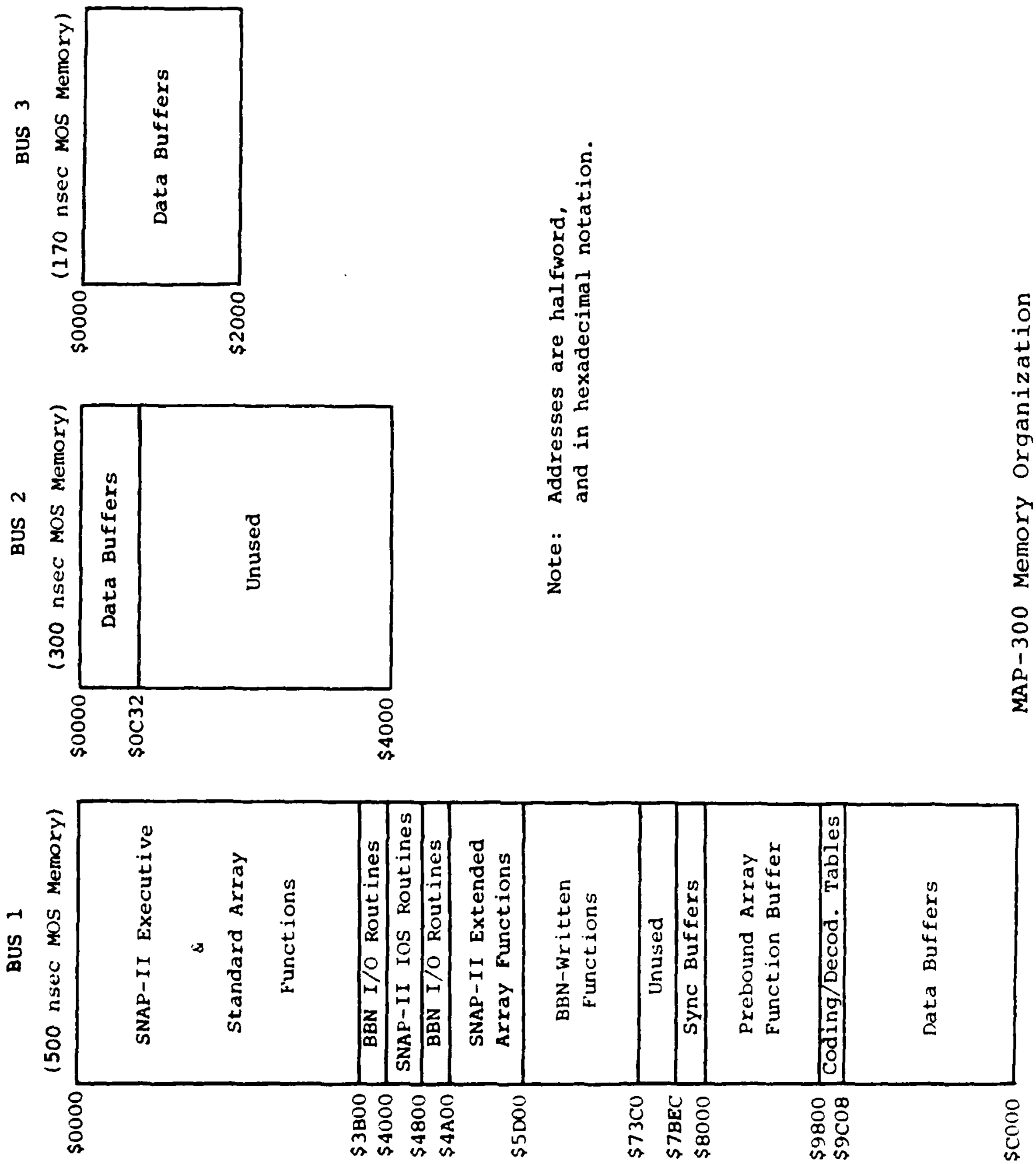
Fig. 16 shows the MAP-300 memory organization, including the location of CSPI and BBN software, as well as the location of buffers defined in the speech coder system.

2.4.1.1 CSPI-Supplied MAP-300 Software

The following CSPI-supplied MAP-300 software modules are required for speech coder operation:

SNAP-II Software System Release 3.5 Model 8300-RSX11M.
(This package includes the SNAP-II Executive and the Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package Release 0.1
Model 8400-RSX11M.
(This package includes the SNAP-II IOS Modules.)



Note: Addresses are halfword, and in hexadecimal notation.

MAP-300 Memory Organization

Fig. 16

These software modules are described in CSPI documentation [2,3,5].

2.4.1.2 BBN-written MAP-300 Software

BBN-written additions and modifications to the SNAP-II software system are contained in four separate files. (For all BBN-written MAP-300 modules, a file extension of .MSO designates "MAP source", .MOB designates "MAP object", and .MLI designates "MAP listing".) The files with first name "BBN16M" contain added SNAP-II functions (array and non-array) for algorithmic and system support processing. "BBN16T" contains tables for quantization and decoding. "BBN16U" contains the error protection and correction modules, ADAM, AOM, and IOS-2 programs, and CSPU routines that respond to interrupts from these devices. These programs and routines are described in Section 2.2 of this report. "BBN16P" contains patches to Release 3.5 of the SNAP-II executive. Assembly-listings of these four files appear in Appendix D of this report.

BBN-written SNAP-II functions are functionally described in Appendix A of this report. Each function has been assigned a Function Control Block (FCB) number. An entry has been made for each new FCB in the Function Dispatch Table (FDT), indicating the location of the APU, APS and/or CSPU module(s) that implement the associated function.

Several unused CSPI-supplied functions, normally available

with Release 3.5 of the SNAP-II software system, have been disabled to provide room in the FDT for BBN-written functions. The following SNAP-II functions are unavailable once the BBN-written MAP-300 software has been loaded.

CXMUL	(FCB# 182)	CMML	(FCB# 220)
CXMUL	(FCB# 183)	CMINV	(FCB3 224)
CSMAI	(FCB# 188)	MWLD	(FCB# 225)
CSMA2	(FCB# 189)	ADMRB	(FCB# 229)
FF2D	(FCB# 212)	VHIST	(FCB# 230)
		VRAN1	(FCB# 235)

These functions can be made available by reloading the SNAP-II software system without loading BBN-written MAP-300 software.

In addition, the standard SNAP-II interrupt routines for interrupts from Device 16 (Lines 1 and 2), Device 22 (Line 1), and Device 23 (Line 1) have been modified to transfer control to BBN-written interrupt routines. The standard interrupt routines for these devices can be similarly accessed by reloading the SNAP-II Executive without loading BBN-written MAP-300 software.

2.4.2 PDP-11 Software Components

The proper operation of the speech coder is defined and controlled by a group of programs running in the PDP-11. These programs include a set of SNAP-II support subroutines and a MAP-300 driver, supplied by CSPI, as well as a number of FORTRAN programs written by BBN that make use of the CSPI-supplied routines to perform the specific task of defining and controlling the operation of the speech coder in the MAP.

CSPI-supplied PDP-11 programs are described in Section 2.4.2.1. PDP-11 programs written by BBN are described in Section 2.4.2.2.

2.4.2.1 CSPI-Supplied PDP-11 Software

The following CSPI-supplied PDP-11 software modules are required for speech coder operation:

SNAP-II Software System Release 3.5 Model 8300-RSX11M. (This package includes the SNAP-II Host Support Packages for Standard and Extended Array Functions.)

SNAP-II Input/Output Scroll Package Release 0.1 Model 8400-RSX11M
(This package includes the SNAP-II IOS Host Support Package.)

DEC RSX-11M I/O Driver Model 8901

These software modules are described in CSPI documentation [2,3,5,6,7].

2.4.2.2 BBN-written PDP-11 Software

A set of BBN-written FORTRAN programs performs the multiple tasks of defining and initializing MAP-300 buffers and scalars and of defining and executing the sequence of functions in the MAP that perform the actual speech coding operations. These FORTRAN programs are organized around a single mainline program (BBN16R), which calls, in turn, subroutines to configure MAP buffers and scalars (BBN16C), initialize these buffers and scalars (BBN16I), define function lists for various speech coder

tasks (BBN16F), and interact with the user in controlling the execution of these function lists (DCA96E). These subroutines are described in Sections 2.4.2.2.1 through 2.4.2.2.4 below.

2.4.2.2.1 Buffer and Scalar Configuration (BBN16C)

The task of defining MAP-300 buffers and scalars to the SNAP-II executive is performed by subroutine BBN16C. Upon entry, this routine first initializes the SNAP-II executive via the MPOPN function. All SNAP-II buffers are then configured using the MPCLB function, with buffer ID numbers, sizes, and addresses defined symbolically within the BBN16C routine. In general, transmitter buffer names begin with "T", while receiver buffer names begin with "R". Buffer sizes are stored in variables named by prefixing an "S" to the the buffer name. Similarly, buffer addresses are stored in variables named by prefixing "A" to the buffer name. The buffers used in the speech coder system are listed in Appendix B. All real scalars and integer scalars are then defined by assigning scalar ID numbers to the associated symbolic scalar names. Transmitter scalar names generally begin with "T", while receiver scalar names begin with "R". Real and integer scalars used in the speech coder system are listed in Appendix C.

All buffer and scalar names (and certain buffer sizes and addresses) are included in FORTRAN labeled COMMON blocks, which permit all other FORTRAN subroutines in the speech coder system

to reference these symbolic buffer and scalar names in their calls to SNAP-II functions.

2.4.2.2.2 Buffer and Scalar Initialization (BBN16I)

MAP-300 buffers and scalars are set to their initial values by subroutine BBN16I. Only certain buffers and scalars require such initialization. Appendices B and C indicate the proper initial contents of MAP buffers, real scalars, and integer scalars.

File "BBN16I" also contains several FORTRAN subroutines called by DCA96I and used to calculate the initial contents of certain buffers. Included are subroutines that generate a square-wave of a given frequency and amplitude (SQWAV) and a Hamming window of a given size (HAMMNG).

2.4.2.2.3 Control Structure Definition (BBN16F)

Subroutine BBN16F defines several function lists that specify the operation of the speech coder system. Function list ID numbers are defined symbolically. The FORTRAN variables containing the function list ID numbers are included in a labeled COMMON block so that they can be referenced by other subroutines.

As part of the function list definition task, subroutine BBN16F produces pre-bound versions of all of the SNAP-II array functions used in the speech coder system. Pre-binding is a SNAP-II operation that causes function parameter information,

normally communicated to the function at execution time, to be "bound" to the function at some earlier time (in this case at system initialization time). Approximately 6100 half-words are required on Bus 1 to store the pre-bound versions of the speech coder array functions.

Function lists ANLZA, ANLZB, SYNZA, and SYNZB are defined for the analysis and synthesis sections of the speech coder. (Since the system is double-buffered, the analysis and synthesis function lists are each defined twice to allow for the different sets of input and output buffers.) These function lists specify the sequence of MAP-300 array and non-array functions, operating on the system buffers and scalars previously defined, that implement the analysis and synthesis processes described in Sections 2.2.1.3 and 2.2.2.3 of this report.

The analysis and synthesis function lists described above are used in the specifications of several other function lists that define the outer structure of the speech coder system. Function list BBNLP defines the real-time speech coder loop, executing function lists ANLZA, SYNZA, ANLZB, and SYNZB when buffer status flags indicate that the input buffers are full and the output buffers are empty for each of these processes. Function list BBNRTS defines the start-up sequence for the real-time speech coder system, starting the Modem, ADAM, and AOM Scroll processors, and repeatedly executing the BBNLP function

list described above. This repeated execution is conditioned on the contents of MAP integer scalar RUN being non-zero; hence, the speech coder can be stopped by setting RUN to zero. Function list BBNRST restarts the real-time speech coder by setting RUN to non-zero and reinitiating the repeated execution of the real-time speech coder loop.

Two other function list definitions are included in this subroutine to support non-real-time file-to-file speech coder operation, speech coder timing operation, and speech coder oscilloscope display operation (for detailed internal timing). These modes of speech coder operation are not supported, and can not be invoked, in the delivered speech coder system. They were used for system development and debugging use under the RT-11 operating system and are included here to aid in future additions to the speech coder system. Function list BBNFFT specifies the speech coder loop with explicit calls to functions that execute the A/D, RMODEM, TMODEM, and D/A interrupt routines. Function list BBNTIM repeatedly invokes a sequence of six BBNFFT function list executions (one for each possible combination of input and output buffers), with software simulation of the transmitter-to-receiver communication path occurring after each BBNFFT execution.

All SNAP-II functions are called via PDP-11 support subroutines that perform the actual communication with the MAP

driver. These subroutines are contained in library SNPLIB for CSPI-supplied functions and in file BBN16H for BBN-written functions.

2.4.2.2.4 System Software Execution (BBN16E)

The execution of the speech coder system is controlled by subroutine BBN16E. This subroutine starts the real-time speech coder system by first loading the Modem, ADAM, and AOM Scrolls with their respective programs, and then executing the real-time speech coder start-up function list (BBNRTS) described in the preceding section. It then interacts with the user, responding to single-character commands to halt the speech coder ('Q'), enable/disable error simulation ('E'), enable/disable error correction ('C'), cause the speech coder to lose sync artificially ('L'), type out a group of speech coder state scalars ('T'), and suspend the controlling RSX-11M task, allowing the speech coder to continue executing in the MAP-300 ('S').

Provision is included in this subroutine for the user to select other speech coder modes of operation, specifically file-to-file, timing, and oscilloscope display modes. However, this mode selection process is bypassed in the delivered speech coder system, and real-time speech coder execution mode is forced. As described in the previous section, these other operating modes are not supported, and cannot be invoked, in the delivered speech coder system. They were used for system development and

debugging and are included here to aid in future additions to the speech coder system. File BBN16D contains dummy versions of various system-dependent subroutines, which constitute the unsupported software portions of these unsupported modes of speech coder operation.

2.5 SYSTEM TIMING PERFORMANCE

The speech coder system introduces a total delay of 9 frames, corresponding to 293.6 milliseconds, between voice input and synthesized voice output (not including any delays in transmission). This consists of 4 frames of delay introduced by the transmitter portion of the system and 5 frames of delay contributed by the receiver portion. In the transmitter, a frame of delay each is produced by the buffering in the ADAM scroll program, the ANALYZE processing module, and the TMODEM scroll program. In addition, the ANALYZE module introduces a frame of "folding" delay due to the concurrent execution of the ANALYZE module and the PROPAR/PRORES modules (see Section 2.2.1).

In the receiver, two frames of delay are contributed by the buffering in the RMODEM scroll program, and a frame each is produced by buffering in the SYNTHESIZE processing module and the AOM scroll program. In addition, the SYNTHESIZE module introduces a frame of "folding" delay due to the concurrent execution of the SYNTHESIZE module and the CORPAR and DECODA/B modules (see Section 2.2.2).

The observed internal timing of the speech coder modules is shown in Fig. 17. This figure reflects data obtained by running the speech coder in a loop without the ADAM, AOM, and MODEM scroll processors and observing the states of the MAP-300 RA flag (APU run) and G-flags with an oscilloscope. The speech coder was provided with simulated A/D input data. ADAM, AOM, and MODEM scroll interrupt routines were omitted from the timing loop. These interrupts were timed separately, and their execution times are indicated at the end of the figure. Including these interrupt routines, Fig. 17 shows a total processing time of 30.0 milliseconds for a 32.625 millisecond frame of data, or about 0.92 times real time. (This is the worst case total processing time. If the ADAM, AOM, or MODEM scroll interrupts were to occur while the AP were running and the CSPU were idle, some or all of the interrupt routine execution time would be hidden behind the concurrent AP routine.)

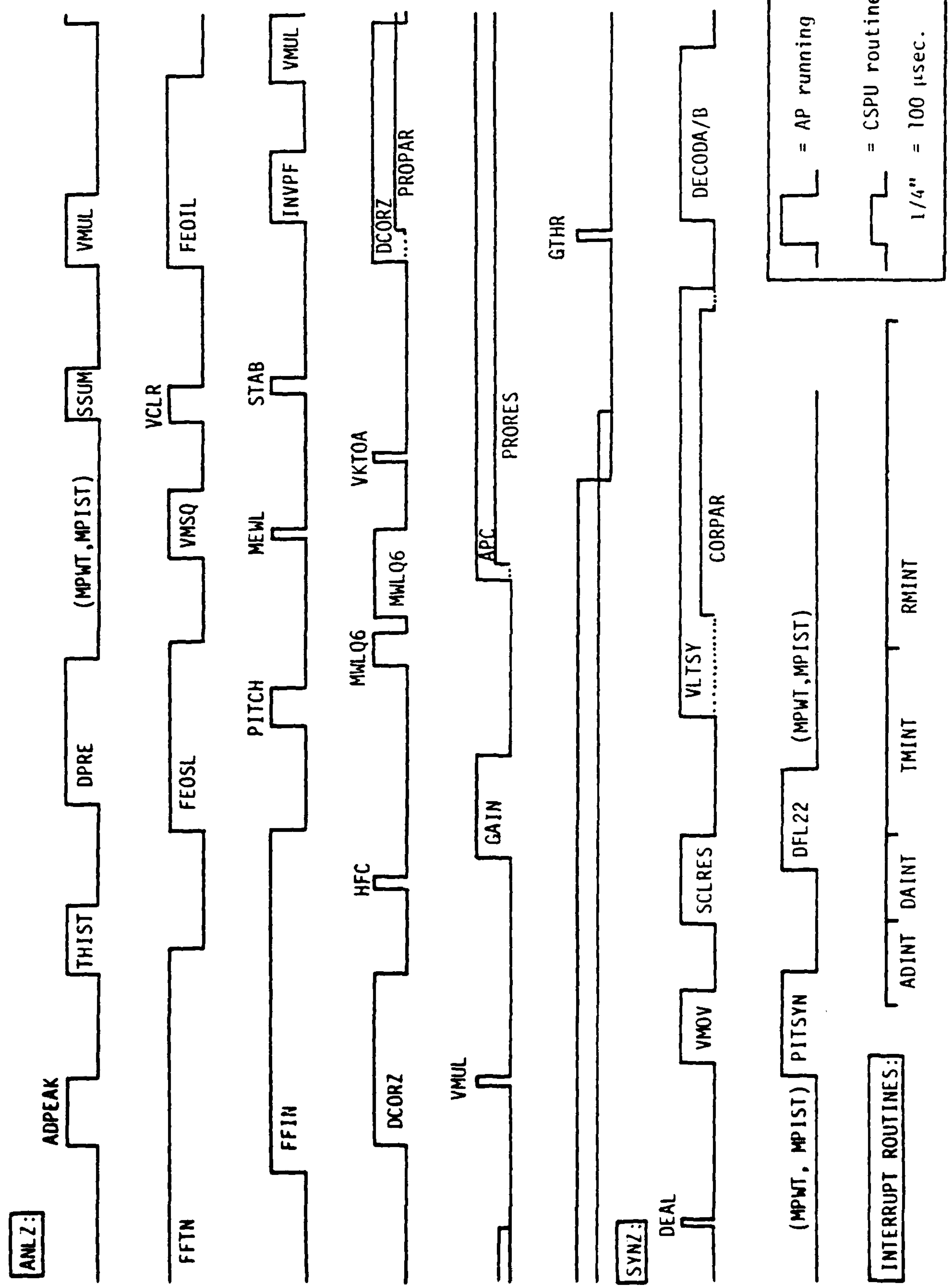


Figure 17 MAP-300 Timing

3. SOFTWARE OPERATING PROCEDURES

The installation and execution of the speech coder system software are accomplished using the RSX-11M operating system as well as MAP-300 support software supplied by CSPI. It is assumed that the RSX-11M operating system, the MAP-300 device drivers, the MAP-300 loaders, and the SNAP-II software system have all previously been installed prior to the speech coder system installation.

After the speech coder system software has been installed, the speech coder can be executed by using the procedure given in Section 3.1. The software installation procedure is given in Section 3.2.

3.1 SOFTWARE EXECUTION PROCEDURE

The speech coder system execution procedure consists of two major steps. First, both MAP-300 processors in the full-duplex speech coder system must be loaded from the host PDP-11 with identical copies of the speech coder MAP-300 software. Second, two RSX-11M tasks must be run to initialize and start the two MAP-300 speech coders.

The following files must be disk-resident before speech coder execution can be attempted:

BBN16X.MBN (MAP Binary file)

```
BBA16.TSK          (Host Task for MAP #A)
BBB16.TSK          (Host Task for MAP #B)
```

If these files do not exist, they must be created according to the speech coder system software installation procedure given in Section 3.2.

The following dialog indicates the proper command sequence for full-duplex speech coder system execution. User commands are underscored to differentiate them from computer responses.

```
>SET /UIC=[6,260]
>ALLOCATE MA:           **allocate MAP #A**
>ALLOCATE MB:           **allocate MAP #B**
>INSTALL BBA16
>INSTALL BBB16
>RUN [6,100]QLA         **load MAP #A**
*BBN16X.MBN
TT1 -- STOP

>RUN [6,100]QLB         **load MAP #B**
*BBN16X.MBN
TT1 -- STOP

>RUN BBA16              **start MAP #A**
BBN 16K BPS SPEECH CODING SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS...
EXECUTING BBN16 SPEECH CODER...
SPEECH CODER IS IN OPERATION.

COMMANDS ARE:
 S:   SUSPEND TASK (LEAVING SPEECH CODER RUNNING)
 Q:   QUIT (HALT SPEECH CODER)
 E N: SIMULATE N ERRORS PER FRAME
 C N: ERROR-CORRECT IF N=0
 L:   CAUSE RCVR TO LOSE SYNC
 T:   TYPE OUT SPEECH CODER STATE

ENTER COMMAND: S
BBA16 -- PAUSE (SPEECH CODER CONTINUING)
```

```
>RUN BBB16              **start MAP #B**
BBN 16K BPS SPEECH CODING SYSTEM

CONFIGURING MAP BUFFERS AND SCALARS...
INITIALIZING MAP BUFFERS AND SCALARS...
PREBINDING MAP FUNCTIONS AND DEFINING FUNCTION LISTS...
EXECUTING BBN16 SPEECH CODER......
SPEECH CODER IS IN OPERATION.

COMMANDS ARE:
 S:   SUSPEND TASK (LEAVING SPEECH CODER RUNNING)
 Q:   QUIT (HALT SPEECH CODER)
 E N: SIMULATE N ERRORS PER FRAME
 C N: ERROR-CORRECT IF N=0
 L:   CAUSE RCVR TO LOSE SYNC
 T:   TYPE OUT SPEECH CODER STATE

ENTER COMMAND: S
BBB16 -- PAUSE (SPEECH CODER CONTINUING)

>
```

At this point in the command sequence, the full-duplex speech coder system should be in operation. Host tasks BBA16 and BBB16 can be resumed (for interaction with the speech coder software) by invoking the RSX-11M "RESUME" command. The speech coder system can be halted either by responding with "Q" to the "ENTER COMMAND:" typeout from BBA16 and BBB16 or by aborting the BBA16 and BBB16 tasks through the use of the RSX-11M "ABORT" command.

3.1.1 Optional Software Execution Procedure

Throughout the BBN16 Speech Coder MAP software, we have included code to set and clear various general-purpose flags (G-flags). These flags are used mostly for debugging, timing, and system measurement purposes; the manipulations are not required,

in general, for speech coder operation. Although these G-flag manipulations are strictly legal and proper according to CSPI documentation, they seem to induce "Bus 1 Memory Timeout" errors after apparently random amounts of time (ranging from a few minutes to several hours). For this reason, we have delivered with the system a patch file (NOGF16.MOB) which, when loaded into the MAP over the BBN16 speech coder software, no-op's all G-flag manipulations not needed for speech coder operation. This file can be loaded into the MAP processors (following the load of BBN16X) using the following command sequence:

```
>RUN [6,100]MALD
OBJECT INPUT? NOGF16.MOB
BINARY OUTPUT?(carriage return)
TT1 -- STOP 2
>RUN [6,100]MBLD
OBJECT INPUT?NOGF16.MOB
BINARY OUTPUT?(carriage return)
TT1 -- STOP 2
>
```

3.1.2 Analog Input Level Setting

The BBN16 speech coder software includes a facility for monitoring the peak input speech level. This facility allows the level to be set properly when the speech coder is being fed from an adjustable level signal source (such as a tape deck). The peak input level is maintained in integer scalar TADPK. TADPK can be displayed (along with a number of other scalars) during speech coder operation by entering a 'T' in response to the 'COMMAND:' prompt. The peak input level range is 0-2047,

so the desired peak input level is approximately 1500-2000. A peak level above about 2000 introduces a potential input signal clipping condition, while a peak level below about 1500 does not take full advantage of the system's available dynamic range. The 'T' command resets TADPK to 0, so the displayed value pertains only to the time interval since the previous 'T' was typed.

3.2 SOFTWARE INSTALLATION PROCEDURE

The speech coder system software installation procedure consists of two major steps. First, the speech coder MAP-300 software must be converted from MAP object format to MAP binary format. Second, two RSX-11M tasks (for initializing and starting the two MAP-300 processors) must be task-built.

The following files must be disk-resident before speech coder software installation can be attempted:

```
BBN16X.MOB              (MAP object file)
             (This file consists of the following
             concatenated files, with intervening
             "FFFF" lines deleted:
                         S300EX.MOB
                         EAF300.MOB
                         IOS300.MOB
                         BBN16M.MOB
                         BBN16U.MOB
                         BBN16T.MOB
                         BBN16P.MOB)
BBN16C.FOR           (HOST FORTRAN module)
BBN16D.FOR           (        "          )
BBN16E.FOR           (        "          )
BBN16F.FOR           (        "          )
BBN16H.FOR           (        "          )
BBN16I.FOR           (        "          )
```

BBN16R.FOR (")

If these files do not exist, they must be recopied to disk from the RSX PIP-format "BBN 16 kb/s speech coder s/w" magnetic tape delivered with the speech coder system.

The MAP-300 speech coder software should be converted from MAP object format to MAP binary format using the following command sequence:

```
>RUN [6,100]MALD
OBJECT INPUT?BBN16X.MOB
BINARY OUTPUT?BBN16X.MBN
LOAD MAP?(Y OR N) N
TT1 -- STOP 2
>
```

The two RSX-11M tasks (for initializing and starting the two MAP-300 processors) should be generated next. First, all FORTRAN speech coder modules listed above must be compiled. Then the two tasks must be task-built, with each task including the compiled FORTRAN object modules, the CSPI-supplied SNAP host support library modules, and the appropriate CSPI-supplied MAP device driver for MAP #A or MAP #B. The following command sequence will accomplish this procedure:

```
>F4P BBN16C=BBN16C.FOR/CO:35
>F4P BBN16D=BBN16D.FOR/CO:35
>F4P BBN16E=BBN16E.FOR/CO:35
>F4P BBN16F=BBN16F.FOR/CO:35
>F4P BBN16H=BBN16H.FOR/CO:35
>F4P BBN16I=BBN16I.FOR/CO:35
>F4P BBN16R=BBN16R.FOR/CO:35
>TKB                      **build host task for MAP #A**
```

```
TKB>BBA16/FP/CP=BBN16R,BBN16C,BBN16D
TKB>BBN16E,BBN16F,BBN16H,BBN16I
TKB>[6,100]SNALIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=11
TKB>ASG=TI:7
TKB>ASG=MA:10
TKB>//
>TKB                   **build host task for MAP #B**
TKB>BBB16/FP/CP=BBN16R,BBN16C,BBN16D
TKB>BBN16E,BBN16F,BBN16H,BBN16I
TKB>[6,100]SNBLIB/LB
TKB>/
ENTER OPTIONS:
TKB>UNITS=11
TKB>ASG=TI:7
TKB>ASG=MB:10
TKB>//
>
```

The speech coder system software is now installed. The speech coder can be executed by using the procedure given in Section 3.1.

4. REFERENCES

1. R. Viswanathan, J. Wolf, L. Cosell, K. Field, A. Higgins and W. Russell, "Design and Real-time Implementation of a Baseband LPC Coder for Speech Transmission over 9600 BPS Noisy Channels," Final Report, BBN Report No. 4327, Vols. I and II, Bolt Beranek and Newman Inc., Cambridge, MA, February 1980.

2. "Simple Notation for Array Processing, Version II (SNAP-II) Reference Manual," CSPI Document No. JB6000-017-01, C.S.P. Inc., Billerica, MA, November 1978.

3. "Release Note: SNAP-II Release 3 Executive Features and Expanded Array Function Set," CSPI Document No. DW6000-021-04, C.S.P. Inc., Billerica, MA, July 1979.

4. "Release Note: SNAP-II Extended Array Function Library, II," CSPI Document No. RB6000-24-00, C.S.P. Inc., Billerica, MA, March 1980.

5. "SNAP-II Input/Output Scroll Package, User's Manual, CSPI Document No. JW8400-001-01, C.S.P. Inc., Billerica, MA, July 1979.

6. "Installation Procedure for Release 3 SNAP-II Software System on the DEC RSX-11M System," CSPI Document No. LL8901-004-03, C.S.P. Inc., Billerica, MA, July 1979.

7. "MAP Software Interface Description for the DEC RSX-11M System," CSPI Document No. ST8901-000-04, C.S.P. Inc., Billerica, MA, July 1979.

APPENDIX A
BBN-WRITTEN FUNCTION DESCRIPTIONS

This Appendix contains a complete user-level description for each MAP-300 SNAP-II function written by BBN and used in the speech coder implementation. The descriptions are ordered alphabetically by function name.

Functions that were used in the speech coder implementation and are not described here were supplied by CSPI as part of the SNAP-II Software System (see Sections 2.4.1.1 and 2.4.2.1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADINT
NAME EXPANSION: Simulate A/D interrupt

FCB #: 124
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: ADAMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

This SNAP callable function calls the ADAMINT A/D Interrupt Service module in exactly the same manner as it is used to respond to A/D interrupts, so it may be used to simulate such an interrupt. See Section 2.2.1.2.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: ADPEAK (U)
NAME EXPANSION: Monitor peak signal from the A/D

FCB #: 182
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: ADPKU
APS MODULE NAME: ADPKA
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer U	1-63	Real	Input

NUMBER OF OUTPUT SAMPLES: 1, in integer scalar TADPK
NOTES: --

FUNCTION DESCRIPTION:

Maintains a running peak absolute value of the data in the input buffers given to it on successive calls. The running peak is maintained as an integer value (to facilitate monitoring the peak level) in the integer scalar TADPK. Since SNAP-II standard binding does not support binding an integer scalar to an array function, the address of TADPK is "hard bound" in the APS module source code.

The input buffer is assumed to contain signals in the range -1 to +1 from a 12-bit A/D converter. The integer value maintained in TADPK falls in the range 0-2047.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: APC(Y,A,U,V,W,R,S,T)
NAME EXPANSION: APC loop

FCB #: 199
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: APCU$
APS MODULE NAME: APCS$
CSPU MODULE NAME: SBM$APC

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Fixed	Output: Coded residual samples
Real Scalar A	0-191	Real	Input: Pitch lag
Buffer U	1-63	Real (YBS)	Input: Preemphasized speech
Buffer V	1-63	Real (15)	Input: Filter coeffs: 6 spectral, 3 pitch, 6 noiseshaping.
Buffer W	1-63	Real (6)	Input: Quantizer segment scale factors, alternating with their reciprocals.
Buffer R	1-63	Real (432)	Input/Output: Pitch Filter History: last frame, current frame.
Buffer S	1-63	Real (6)	Input/Output: Spectral Filter History
Buffer T	1-63	Real (6)	Input/Output: Noise Shaping Filter History

NUMBER OF OUTPUT SAMPLES: YBS + RBS/2 + SBS + TBS
NOTES: Uses SA to modify APS code.
Requires special CSPU binding.

FUNCTION DESCRIPTION:

Compute and code APC residual. See Section 2.7 of the Algorithm Specification (Volume I, Appendix A of this report) for complete description.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: CORPAR(I)
NAME EXPANSION: Correct parameters

FCB #: 123
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: CORPAR

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal I	-2,0	Integer	Specifies buffer pair

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Unbitstreams, error-corrects, and decodes parameters from on of the RBITS buffers to one of the RSOURCE buffers. The tables used for decoding each parameter are given in the file BBN16T.MSO.

The argument I specifies which pair of RBITS/RSOURCE buffers is used. I=-2 means RBTA/RSRA, and I=0 means RBTB/RSRB.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DAINT
NAME EXPANSION: Simulate D/A Interrupt

FCB #: 127
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: AOMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

The SNAP-callable function calls the AOMINT D/A Interrupt Service module, so it may be used to simulate such an interrupt. See Section 2.2.2.5.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DEAL(Y,A,U,B,V,W)
NAME EXPANSION: Deal received decoded data from single buffer

FCB #: 212
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DEALU$
APS MODULE NAME: DEALS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (230)	Input: Received decoded data buffer
Real Scalar A	0-191	Real	Output: Received $\sqrt{\text{Energy}}$
Buffer U	1-63	Real (3)	Output: Received $\sqrt{\text{Delta Gains}}$
Real Scalar B	0-191	Real	Output: Received Pitch Lag
Buffer V	1-63	Real (3)	Output: Received Pitch Predictor Coeffs
Buffer W	1-63	Real (6)	Output: Received Spectral Predictor Coeffs

NUMBER OF OUTPUT SAMPLES: 14
NOTES:

FUNCTION DESCRIPTION:

On function exit, Y(Ø - 215) is expected to contain 216 received residual samples.

```
     SA  <= Y(216)
U(Ø-2)<= Y(217-219)
     SB  <= Y(220)
V(Ø-2)<= Y(221-223)
W(Ø-5)<= Y(224-229)
```

AD-A096 092 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1

DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)

DEC 80 J J WOLF, K D FIELD, W H RUSSELL DCA100-79-C-0037

UNCLASSIFIED BBN-4565-VOL-2 NL

2 OF 5

AD A
096092

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DECODA
NAME EXPANSION: Decode residual, buffer-pair "A"

FCB #: 188
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DCDU
APS MODULE NAME: DCDSA
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Unbitstreams and decodes the residual samples from buffer RBTA to buffer RSRA. The table used for decoding the residual is given in the file BBN16T.MSO.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: DECODB
NAME EXPANSION: Decode residual, buffer-pair "B".

FCB #: 189
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: DCDU
APS MODULE NAME: DCDSB
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Unbitstreams and decodes the residual samples from buffer RBTB to buffer RSRB. The table used for decoding the residual is given in the file BBN16T.MSO.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: FFILR(Y,M,U,V,W)
NAME EXPANSION: Fast Fourier Transform, Not-in-Place, To Real Data, Long

FCB #: 198, 206
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: --
APS MODULE NAME: -- (see below)
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS

(see below)

NUMBER OF OUTPUT SAMPLES: USB*2
NOTES: One complex sample is input beyond the end of the configured U buffer (see below).

FUNCTION DESCRIPTION:

This is a slightly modified version of the CSPI function FFINR (from SNAP-II, Extended Array Function Library II [4]). The modification involves an assumed reformatting of the complex frequency domain input (U) such that the last real input point is assumed to appear as such, rather than in the first imaginary input location (which is known to be zero).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: FFTLR(Y,M,U,V,W)
NAME EXPANSION: Fast Fourier Transform, Not-in-Place, On Real Data, Long

FCB #: 204, 197
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: --
APS MODULE NAME: -- (see below)
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS

(see below)

NUMBER OF OUTPUT SAMPLES: UBS/2 + 1
NOTES: One complex sample is output beyond the end of the configured Y buffer (see below).

FUNCTION DESCRIPTION:

This is a slightly modified version of the CSPI function FFTNR (from SNAP-II, Extended Array Function Library II [4]). The modification involves a reformatting of the complex frequency domain output (Y) such that the last real output point appears as such, rather than in the first imaginary output location (which is known to be zero).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: GAIN(Y,A,U,B,V,C)
NAME EXPANSION: Compute Quantizer Segment Scale Factors

FCB #: 196
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: GAIN$
APS MODULE NAME: GAINS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (3)	Output: Quantizer segment scale factors, alternating with their reciprocals.
Real Scalar A	0-191	Real	Output: Coded frame energy
Buffer U	1-63	Real (3)	Output: Coded segment delta-gains
Real Scalar B	0-191	Real	Input: 1/72
Buffer V	1-63	Real (216)	Input: Second Residual Samples
Real Scalar C	0-191	Real	Input: 1/3

NUMBER OF OUTPUT SAMPLES: 7 total
NOTES: Fixed to process 216 input buffer V elements, in 3 segments of 72 elements each.

FUNCTION DESCRIPTION:

$$SG(n) = \frac{1}{72} \sum_{i=1}^{72} (V(i+(n*72))^2) \qquad n = \emptyset,1,2$$

$$G = \frac{1}{3} \sum_{i=1}^{72} (SG(i)) \; ; \; GH = \text{quantized } \sqrt{G}; \quad SA = \text{coded } G$$

$$DG(n) = SG(n)/(GH*GH) \qquad n = \emptyset,1,2$$

$$U(n) = \text{coded } DG(n) \; ; \; DGH(n) = \text{quantized } \sqrt{DG(n)} \qquad n=\emptyset,1,2$$

$$\left.\begin{array}{l} Y(2n) = (GH*DGH(n)) \\ Y(2n+1) = 1./Y(2n) \end{array}\right\} \quad n=\emptyset,1,2$$

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: GTHR(Y,A,U,B,V,W)
NAME EXPANSION: Gather Transmission data into single buffer

FCB #: 200
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: GTHRU$
APS MODULE NAME: GTHRS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Fixed (230)	Output: Transmission buffer
Real Scalar A	0-191	Fixed (in lefthword)	Input: Coded Energy
Buffer U	1-63	Fixed (3)	Input: Coded Delta Gains
Real Scalar B	0-191	Fixed (in lefthword)	Input: Coded Pitch Lag
Buffer V	1-63	Fixed (3)	Input: Coded Pitch Predictor Coeffs.
Buffer W	1-63	Fixed (6)	Input: Coded Spectral Predictor Coeffs.

NUMBER OF OUTPUT SAMPLES: YBS-216
NOTES:

FUNCTION DESCRIPTION:

On function call, Y(0-215) is expected to contain 216 coded residual samples.

```
Y(216)      <=  SA
Y(217-219)  <=  U(0-2)
Y(220)      <=  SB
Y(221-223)  <=  V(0-2)
Y(224-229)  <=  W(0-5)
```

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: HFC(Y,U,V)
NAME EXPANSION: High Frequency Correction

FCB #: 190
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: HFCU$
APS MODULE NAME: HFCS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: HFC'd autocorrelation coeffs
Buffer U	1-63	Real	Input: Autocorrelation coeffs
Buffer V	1-63	Real (3)	Input: LAMBDA*MU(i) LAMBDA = .035 MU = $\begin{pmatrix} .375 \\ -.25 \\ .0625 \end{pmatrix}$

NUMBER OF OUTPUT SAMPLES: YBS
NOTES:

FUNCTION DESCRIPTION:

For i = Ø,2:
Y(i) = U(i) + E2*V(i)

For i = 2, (YBS-1):
Y(i) = U(i)

K1 = U(1)/U(Ø)

E1 = (1 - K1^2)*U(Ø)

K2 = (U(2) - K1*U(1))/E1

E2 = E1*(1 - K2^2)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: INVPF(Y,A,U,V)
NAME EXPANSION: 3-tap Inverse Pitch Filter

FCB #: 167
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: INVPU$
APS MODULE NAME: INVPSI
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: filter output
Real Scalar A	50-191	Real	Input: pitch in sample-times
Buffer U	1-63	Real (YBS+216)	Input: filter input, including previous 216 samples.
Buffer V	1-63	Real (3)	Input: Filter coefficients

NUMBER OF OUTPUT SAMPLES: YBS
NOTES: This routine uses SA to modify its APS code. Buffer U must be compact 32-bit floating point.

FUNCTION DESCRIPTION:

Implements a 3-tap FIR pitch filter. Let M = SA, the pitch in terms of sample times:

$$y(n) = u(n) + \sum_{j=0}^{2} v(j)*u(n - M - j + 1), \quad n=0,1,\ldots YBS-1$$

The pitch can take on values $14 \leq M \leq 133$, so the buffer U contains the "previous" frame of 216 samples in addition to the "current" frame. That is, U(o) = u(-216), U(216) = u(o), etc.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPGSC(GFLAG,SETCLR)
NAME EXPANSION: G-Flag Set/Clear

FCB #: 106
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPGSC

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal flag	0-3	Integer	Selects G-flag
Literal SETCLR	0-1	Integer	SETCLR=0=>Set flag SETCLR=1=>Clear flag

NUMBER OF OUTPUT SAMPLES: -
NOTES:

FUNCTION DESCRIPTION:

MPGSC sets or clears one of the four G-flags. This is useful for program timing using an external oscilloscope.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MPIFF(IA,IB, FLID)
NAME EXPANSION: If (IA.NE.0)& (IB.EQ.0) Conditional Function List Execution

FCB #: 105
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: -
APS MODULE NAME: -
CSPU MODULE NAME: MPIFF$

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Integer Scalar IA	0-127	Integer	Input
Integer Scalar IB	0-127	Integer	Input
Literal FLID	0-63	Integer	Input: Function List ID

NUMBER OF OUTPUT SAMPLES: -
NOTES: Function list 'FLID' must be previously defined.

FUNCTION DESCRIPTION:

Function list 'FLID' is executed if and only if Integer Scalar IA is not equal to zero, and Integer Scalar IB is equal to zero.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: MWLQ6(Y,A,U,V,W)
NAME EXPANSION: Matrix (Weinter-Levinson-Durbin) Solution, with Quantized and coded outputs, 6th order
FCB #: 135
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: MWLF$APU
APS MODULE NAME: MWLF$APS
CSPU MODULE NAME: MWLQ$SSM

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (12)	Output: Reflection coefficients and error terms (as in MWLD)
Real Scalar A	0-191	Real	Input: Threshold value (as in MWLD)
Buffer U	1-63	Real (6)	Output: Quantized reflection coefficients
Buffer V	1-63	Real (7)	Input: Autocorrelation coefficients (as in MWLD).
Buffer W	1-63	Long Fixed (6)	Output: Coded reflection coefficients.

NUMBER OF OUTPUT SAMPLES: 6
NOTES: Special intermediate CSPU support is used.

FUNCTION DESCRIPTION:

This module consists of the SNAP-II function MWLD followed by a special quantization/coding module. MWLD outputs LPC coefficients into the buffer U, but the Q/C module overwrites them with the quantized reflection coefficients. The threshold and quantized-value tables are not specified as parameters, but are referenced symbolically by the APS module.

MWLD requires that buffers Y,U,V be compact (SI=1). Buffer W must be compact also.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PITCH(Y,A,U,B,C,D)
NAME EXPANSION: Compute and code pitch

FCB #: 134
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PITU$
APS MODULE NAME: PITS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (3)	Output: Autocorrelation coefficients -RP(Pitch lag -1) -RP(Pitch lag) -RP(Pitch lag +1)
Real Scalar A	0-191	Real	Output: Pitch lag (in samples)
Buffer U	1-63	Real (512)	Input: Autocorrelation coefficients
Real Scalar B	0-191	Fixed (in lefthword)	Output: Coded pitch
Real Scalar C	0-191	Real	Input: Constant 14.
Real Scalar D	0-191	Real	Output: Single tap coefficient

NUMBER OF OUTPUT SAMPLES: 3 Output Scalars, 3 Output Buffer samples
NOTES: Uses modified SMAX.
Only uses part of input buffer U.
Uses SA to modify APS code
FUNCTION DESCRIPTION:

A = index of $\max\limits_{i=14}^{133}\ [U(i)]$; where first U element is U(Ø).

left half-word of B = A - 14. = A - C

(This value is generated by multiplying by 2**-15, and ALIGN'ing.)

$D = \dfrac{-U(\text{pitch lag})}{U(\emptyset)}$ (single tap coefficient (for case of 3-taps unstable))

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PITSYN(Y,A,U,V)
NAME EXPANSION: 3-tap direct form pitch-synthesis recursive filter

FCB #: 203
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: PITSU$
APS MODULE NAME: PITSSI
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE (length)	COMMENTS
Buffer Y	1-63	Real (UBS+216)	Output: filter output plus 216 preceding output samples.
Real Scalar A	50-191	Real	Input: Pitch in samples
Buffer U	1-63	Real	Input: Filter input
Buffer V	1-63	Real (3)	Input: Filter Coefficients

NUMBER OF OUTPUT SAMPLES: UBS
NOTES: Uses value of SA to modify the APS code. Buffer Y must be compact 32-bit floating point.

FUNCTION DESCRIPTION:

Implements a 3-pole recursive filter for pitch synthesis. Let M=SA, the pitch expressed in samples

$$y(n) = u(n) - \sum_{j=0}^{2} v(j)*y(n-M-j+1), \quad n=0,1,...UBS-1$$

The pitch can take on values $14 \leq M \leq 133$, so the buffer Y must contain "previous" samples; hence, it functions as both input (previous output) and output (output computed by the function). For buffer-configuration reasons, Y must contain the previous 216 samples of output.

Buffer Y is defined so that the first value in it corresponds to y(-216).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PROPAR(I)
NAME EXPANSION: Protect analysis parameters

FCB #: 122
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: PROPAR

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal I	-2,0	Integer	Specifies buffer pair

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Error protects and bitstreams analysis parameters from one of the TSINK buffers into one of the TBITS buffers; also accumulates histogram statistics for all the analysis parameters except pitch.

The argument I specifies which pair of TSINK/TBITS buffers is used. I=-2 means TSNKA/TBTA, and I=0 means TSNKB/TBTB.

See Section 2.2.1.3.18.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: PRORES(I)
NAME EXPANSION: "Protect" residual samples

FCB #: 121
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: PRORES

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Literal I	-2,0	Integer	Specifies buffer pair

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

Bitstreams coded residual samples from one of the TSINK buffers into one of the TBITS buffers. No "protection" is done.

The argument I specifies which pair of TSINK/TBITS buffers is used. I=-2 means TSNKA/TBTA, and I=0 means TSNKB/TBTB.

See Section 2.2.1.3.18.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: RMINT
NAME EXPANSION: Simulate Receiver Modem Interrupt

FCB #: 126
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: RMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the RMODEMINT Receiver Modem Interrupt Service module, so it may be used to simulate such an interrupt. See Section 2.2.2.2.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: SCLRES(Y,A,U,V)
NAME EXPANSION: Compute Quantizer scale factors and scale residual

FCB #: 202
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: SCL$
APS MODULE NAME: SCLS$
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real (216)	Output: Scaled residual samples
Real Scalar A	0-191	Real	Input: Received $\sqrt{\text{Frame Energy}}$
Buffer U	1-63	Real (3)	Input: Received Delta Gains
Buffer V	1-63	Real	Input: Received Residual samples

NUMBER OF OUTPUT SAMPLES: 216
NOTES:

FUNCTION DESCRIPTION:

GFAC(j) = [SA*U(j)]   j = 1,2,3

Y(i) = V(i)*GFAC(j)   j = 1 for i = 1, 72
                        j = 2 for i = 73, 144
                        j = 3 for i = 145, 216

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: STAB(Y,A,U,V)
NAME EXPANSION: Stability check on pitch predictor coefficients; code and quantize.
FCB #: 150
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: STABS
APS MODULE NAME: STABSS
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Quantized stable pitch predictor coeffs.
Real Scalar A	0-191	Real	Input: Single Tap coeff.
Buffer U	1-63	Fixed	Output: Coded stable pitch predictor coeffs.
Buffer V	1-63	Real	Input: Pitch Predictor coefficients.

NUMBER OF OUTPUT SAMPLES: 3 Quantized, 3 coded values
NOTES: Module fixed to process 3 pitch predictor coeffs.

FUNCTION DESCRIPTION:

```
Do stability check on V(Ø), V(1), V(2):
     T1 = V(Ø) + V(1) + V(2)
     T2 = V(Ø) - 2V(1) + V(2)
     T3 = V(Ø) - V(2)
If [(T1 <-1) .OR. (T2 < -1) .OR. (T3 < -1) .OR. (T2 > +2) .OR. (T3 > +1)]

           then unstable:  set C1 = Ø
                               C2 = SA
                               C3 = Ø

Else, stable:   set  C1 = V(Ø)
                     C2 = V(1)
                     C3 = V(2).

Code and quantize C1, C2, C3, set Y buffer to quantized values,
                              set U buffer to coded values.
```

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: THIST(Y,U,V,W,R,S)
NAME EXPANSION: Set up transmitter frame histories.

FCB #: 201
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: THU$
APS MODULE NAME: THS$
CSPU MODULE NAME: SBMSTH

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: Preemphasized speech: history,current frame
Buffer U	1-63	Real (216)	Input: Preemphasized speech: current
Buffer V	1-63	Real	Output: First residual: history,current frame
Buffer W	1-63	Real (6)	Input: First residual: last 6 elements of current frame
Buffer R	1-63	Real	Output: APC pitch filter history last,current frame
Buffer S	1-63	Real (216)	Input: APC pitch filter history: current frame

NUMBER OF OUTPUT SAMPLES: UBS + WBS + SBS
NOTES: Requires special CSPU binding.

FUNCTION DESCRIPTION:

Y(Ø) - Y(UBS - 1) <= U(Ø) - U(UBS - 1)
V(Ø) - V(WBS - 1) <= W(Ø) - W(WBS - 1)
R(Ø) - R(SBS - 1) <= S(Ø) - S(SBS - 1)

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: TMINT
NAME EXPANSION: Simulate Transmitter Modem Interrupt

FCB #: 125
ARRAY OR NON-ARRAY FUNCTION: Non-array

APU MODULE NAME: --
APS MODULE NAME: --
CSPU MODULE NAME: TMODEMINT

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
None			

NUMBER OF OUTPUT SAMPLES: --
NOTES:

FUNCTION DESCRIPTION:

This SNAP-callable function calls the TMODEMINT Transmitter Modem Interrupt Service module, so it may be used to simulate such an interrupt. See Section 2.2.1.5.

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VKTOA(Y,U)
NAME EXPANSION: Convert Reflection Coefficients (K) to Linear Prediction Coefficients (A)
FCB #: 133
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VKAU$
APS MODULE NAME: VKASI
CSPU MODULE NAME: --

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output: LP coefficients A(1), A(2),..., A(UBS)
Buffer U	1-63	Real	Input: Reflection Coefficients K(1), K(2),...K(UBS)

NUMBER OF OUTPUT SAMPLES: UBS
NOTES: This function is different from VKTOA in the 9.6 kb/s MAP-300 system; that was specific to an 8-pole system; this version is general. Also, this version does not output A(Ø).
FUNCTION DESCRIPTION:

The LPC coefficients A(i), i = Ø,1, ... p are computed from the reflection coefficients K(j), j = 1,2, ... p, by the recursion:

$$A_m(m) = K(m)$$
$$A_m(L) = A_{m-1}(L) + K(m)A_{m-1}(m-L) \quad L = 1, \ldots, m-1 \quad m = 1, \ldots, p$$

where $A_m(L)$ indicates the value of the L-th coefficient on the m-th iteration.

The output buffer does not contain A(0), which is 1.0. The first output value is A(1).

BBN-WRITTEN SNAP-II FUNCTION DESCRIPTION

FUNCTION NAME: VLTSY(Y,U,V,W)
NAME EXPANSION: Lattice Synthesis Filter

FCB #: 132
ARRAY OR NON-ARRAY FUNCTION: Array

APU MODULE NAME: VLTSY$
APS MODULE NAME: V3200$
CSPU MODULE NAME: -

PARAMETER DEFINITIONS:

PARAMETER	RANGE	SAMPLE TYPE	COMMENTS
Buffer Y	1-63	Real	Output synthetic speech samples
Buffer U	1-63	Real(8)	Filter memory (G's)
Buffer V	1-63	Real(8)	Reflection coefficients (K's)
Buffer W	1-63	Real	Input samples

NUMBER OF OUTPUT SAMPLES: WBS
NOTES: Exactly 8 elements from each U and V are used. Contents of U are changed.

FUNCTION DESCRIPTION:

Performs lattice form all-pole filter, using reflection coefficients as filter coefficients.

Implements the following Fortran code:

```
      DO 20 I=1, WBS
      F(7)=W(I)-G(7)*K(8)
      DO 10 J=6,0, -1
      F(J)=F(J+1)-G(J)*K(J+1)
10    G(J+1)=G(J)+F(J)*K(J+1)
      G(0)=F(0)
20    Y(I)=F(0)
```

APPENDIX B
MAP-300 BUFFERS

The table contained in this Appendix describes the characteristics of each data buffer used in the MAP-300 speech coder implementation. The table entry labeled "BID" indicates the buffer identification number (for SNAP-accessible buffers) or is blank (for non-SNAP buffers). The entries labeled "Written by:" and "Read by:" indicate subroutines, function lists, or program modules that so access each buffer. (The A/D, D/A, transmitter modem, and receiver modem interrupt service routines are specified in these tables by "ADINT", "DAINT", "TMINT", and "RMINT", respectively.)

MAP-300 Buffers

Usage key: 1 - constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
CTHC1 } CTHC3 }	-	Coding Table of Pitch Tap Threshs C1, C3	Fixed Address	--	ANLZ (Pitch)	1	RL LNG	8	38912	1	(see BBN16T)	1
CTHC2	--	Coding Table of Pitch Tap Threshs C2	Fixed Address	--	ANLZ (Pitch)	1	RL LNG	16	38928	1	(see BBN16T)	1
CTHK1	--	Coding Table of K(1) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LNG	64	38960	1	(see BBN16T)	1
CTHK2	--	Coding Table of K(2) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LNG	32	39088	1	(see BBN16T)	1
CTHK3	--	Coding Table of K(3) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LNG	16	39152	1	(see BBN16T)	1
CTHK4	--	Coding Table of K(4) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LNG	16	39184	1	(see BBN16T)	1
CTHK5	--	Coding Table of K(5) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LNG	16	39216	1	(see BBN16T)	1

MAP-300 Buffers

Usage key: 1 = constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
CTHK6	--	Coding Table of K(6) Thresholds	Fixed Address	--	ANLZ (Spec)	1	RL LNG	16	39248	1	(see BBN16T)	1
CTHG	--	Coding Table of Energy Thresholds	Fixed Address	--	ANLZ (Gain)	1	RL LNG	64	39280	1	(see BBN16T)	1
CTHDG	--	Coding Table of Delta Gain Thresholds	Fixed Address	--	ANLZ (Gain)	1	RL LNG	4	39408	1	(see BBN16T)	1
CTHU	--	Coding Table of Residual Sample Thresholds	Fixed Address	--	ANLZ (APC)	1	RL LNG	4	39416	1	(see BBN16T)	1
DVLC1 DVLC3	--	Decoding Table of Pitch Tap Values C1, C3 (3bits)	Fixed Address	--	ANLZ (Pitch) CORPAR	1	RL LNG	8	39424	1	(see BBN16T)	1
DVLC2	--	Decoding Table of Pitch Tap Values C2 (4bits)	Fixed Address	--	ANLZ (Pitch) CORPAR	1	RL LNG	16	39440	1	(see BBN16T)	1
DVLK1	--	Decoding Table of Reflection Coefficient Values K(1) (6 bits)	Fixed Address	--	ANLZ (Spec) CORPAR	1	RL LNG	64	39472	1	(see BBN16T)	1

MAP-300 Buffers

Usage key: 1 - constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
DVLK2	--	Decoding Table of Reflection Coefficient Values K(2) (5 bits)	Fixed Address	--	ANLZ (Spec) CORPAR	1	RL LNG	32	39600	1	(see BBN16T)	1
DVLK3	--	Decoding Table of Reflection Coefficient Values K(3) (4 bits)	Fixed Address	--	ANLZ (Spec) CORPAR	1	RL LNG	16	39664	1	(see BBN16T)	1
DVLK4	--	Decoding Table of Reflection Coefficient Values K(4) (4 bits)	Fixed Address	--	ANLZ (Spec) CORPAR	1	RL LNG	16	39696	1	(see BBN16T)	1
DVLK5	--	Decoding Table of Reflection Coefficient Values K(5) (4 bits)	Fixed Address	--	ANLZ (Spec) CORPAR	1	RL LNG	16	39728	1	(see BBN16T)	1
DVLK6	--	Decoding Table of Reflection Coefficient Values K(6) (4 bits)	Fixed Address	--	ANLZ (Spec) CORPAR	1	RL LNG	16	39760	1	(see BBN16T)	1
DVLG	--	Decoding Table of $\sqrt{\text{Energy}}$ Value (6 bits)	Fixed Address	--	ANLZ (Gain) CORPAR	1	RL LNG	64	39792	1	(see BBN16T)	1
DVLDG	--	Decoding Table of $\sqrt{\text{Delta Gain}}$ Values (2 bits)	Fixed Address	--	ANLZ (Gain) CORPAR	1	RL LNG	4	39920	1	(see BBN16T)	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
DVLU	--	*Decoding Table of Residual Samples Values (2 bits)*	Fixed Address	--	ANLZ (APC)	1	RL LNG	4	39928	1	(see BBN16T)	1
DVLUF	--	Folded Decoding Table of Residual Samples	Fixed Address	--	DECODA/B	1	RL LNG	4	39936	1	(see BBN16T)	1
HISTO	--	Histogram Data	Fixed Address	PROPAR	HIST16.SAV	1	RL LNG	160	39944	1	zeros	2
TADBA	--	A/D Input from ADAM	--	ADAM Scroll Program	ADINT	1	RL SHRT	216	40264	1	--	2
TADBB	--	A/D Input from ADAM	--	ADAM Scroll Program	ADINT	1	RL SHRT	216	40480	1	--	2
TADBC	--	A/D "On-hook" Tone Data	--	BBN16T	ADINT	1	RL SHRT	216	40696	1	Approx. 240 Hz Square wave	1
TSRA	1	Input Speech to ANLZA	--	ADINT	ANLZ (Premp)	1	RL SHRT	216	40912	1	--	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TSRB	2	Input Speech to ANLZB	--	ADINT	ANLZ (PREMP)	1	RL SHRT	216	41128	1	--	2
TONES	3	Unity Vector	(Constant vector)	BBN16I	ANLZ (PREMP) (SPEC)	Ø	RL LNG	(216)	Ø	3	Ones (Constant vector)	1
TSPØ	4	Preemphasized speech (current frame)	← 265 → ← 216 → ← 216 → ← TSPØ → ← TSP → ← TSP1 →	ANLZ (PREMP)	ANLZ (PITCH)	1	RL LNG	(216)	434	3	--	2
TSP1	5	Preemphasized speech (last frame and current frame)	(see TSPØ)	ANLZ (PREMP)	ANLZ (PITCH) (INVPF)	1	RL LNG	432	2	3	zeros	2
TSP	6	Preemphasized speech (last 49 of last frame, all 216 of current frame)	(see TSPØ)	ANLZ (THIST) (PREMP)	ANLZ (PITCH)	1	RL LNG	(265)	336	3	--	2
THAMØ	7	Hamming window coefficients for 265 sample window	--	BBN16I	ANLZ (PITCH)	1	RL LNG	265	Ø	2	265 Hamming Window Coefficients	1
TWSP	8	Windowed speech	← 512 → ← 265 → Ø's ← TWSPZ → ← TWSP →	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	(265)	866	3	--	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TWSPZ	9	Windowed speech with appended zeros	(see TWSP)	BBN16I	ANLZ (PITCH)	1	RL LNG	512	866	3	zeros	2/1
TCOST	10	Cosine table for FFT	--	BBN16I	ANLZ (PITCH)	1	RL LNG	256	1890	3	cosine table	1
TXSP	11	Frequency domain representation of TWSPZ (this buffer contains 256 complex points; a 257th complex point will be assumed present directly following it).	Each complex sample: real | imag. TXSPR(i) TXSPI(i)	ANLZ (PITCH)	ANLZ (PITCH)	1	CMPLX LNG	(256)	2402	3	--	2
TXSPR	12	Real part of TXSP, including 257th point	(see TXSP)	ANLZ (PITCH)	ANLZ (PITCH)	2	RL LNG	257	2402	3	--	2
TXSPI	13	Imaginary part of TXSP, including 257th point	(see TXSP)	ANLZ (PITCH	ANLZ (PITCH)	2	RL LNG	257	2404	3	--	2
TPSP	14	Complex buffer of TPSPR, TPSPI (this buffer contains 256 complex points; a 257th complex point will be assumed present directly following it).	Each complex sample: real | imag. TPSPR(i) TPSPI(i)	ANLZ (PITCH)	ANLZ (PITCH)	1	CMPLX LNG	(256)	3430	3	--	2
TPSPR	15	Power spectrum of TXSP comprising real part of TPSP, including 257th point.	(see TPSP)	ANLZ (PITCH)	ANLZ (PITCH)	2	RL LNG	257	3430	3	--	[illegible]

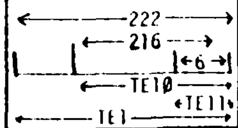
MAP-300 Buffers

Usage key: 1 - constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TPSPI	16	Zeros, comprising imaginary part of TPSP, including 257th point	(see TPSP)	BBN161	ANLZ (PITCH)	2	RL LNG	257	3432	3	zeros	2
TRP	17	Autocorrelation coefficients for pitch determination (IFFT (TPSP))	--	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	512	4458	3	--	2
TC	18	Pitch predictor coeffs.	--	ANLZ (PITCH)	ANLZ (INVPF)	1	RL LNG	3	5482	3	--	2
TRPP	19	Autocorrelation coeffs. -TRP(Pitch lag -1), -TRP(Pitch lag), -TRP(Pitch lag +1)	--	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	3	5488	3	--	2
TWORK	20	Work area for MLWL	--	ANLZ (PITCH)	ANLZ (PITCH)	1	RL LNG	6	5494	3	--	2
TIC	21	Coded pitch predictor coefficients	--	ANLZ (PITCH)	ANLZ (GTHR)	1	FXD LNG	3	5506	3	--	2
TCH	22	Quantized pitch predictor coefficients	1 ← 6 → ← 3 → ← 6 → ←TAH→←TCH→←TANS→ TAHR→ ←TFILT→	ANLZ (PITCH)	ANLZ (INVPF) (APC)	1	RL LNG	3	5524	3	--	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TE1Ø	23	First residual (current frame)		ANLZ (INVPF)	ANLZ (SPEC)	1	RL LNG	(216)	5554	3	--	2
TE11	24	First residual (last 6 samples of current frame)	(see TE1Ø)	ANLZ (INVPF)	ANLZ (INVSF)	1	RL LNG	(6)	5974	3	--	2
TE1	25	First residual (last 6 samples of last frame, plus all of current frame)	(see TE1Ø)	ANLZ (INVPF) (THIST)	ANLZ (INVSF)	1	RL LNG	222	5542	3	zeros	2
THAM1	26	Hamming window coeffs for 216 sample window	--	BBN161	ANLZ (SPEC)	1	RL LNG	216	530	2	216 Hamming Window Coefficients	1
TWF1	27	Windowed First Residual	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	216	962	2	--	2
TRS	28	Autocorrelation coeffs for Spectral Analysis	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	7	5986	3	--	2
TLMU	29	LAMBDA*MU(n) for HFC	--	BBN161	ANLZ (SPEC)	1	RL LNG	3	6000	3	$\begin{pmatrix} .035 \end{pmatrix} * \begin{pmatrix} .375 \\ -.25 \\ .0625 \end{pmatrix}$	1

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TRSP	30	HFC'd Autocorrelation Coeffs.	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	7	6006	3	--	2
TKE	31	Reflection Coeffs. Error terms from MWLQ6	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	12	6020	3	--	2
TKH	32	Quantized Reflection Coeffs.	--	ANLZ (SPEC)	ANLZ (SPEC)	1	RL LNG	6	6044	3	--	2
TIK	33	Coded Reflection Coeffs.	--	ANLZ (SPEC)	ANLZ (GTHR)	1	FXD LNG	6	6056	3	--	2
TAH (TFILT)	34	Predictor Coeffs (also start of filter coeff block)	(see TCH)	ANLZ (SPEC)	ANLZ (SPEC) (NS) (APC)	1	RL LNG	(6)	5512	3	--	2
TAHR	35	Reverse Buffer of TAH with last element = 1.	(see TCH)	ANLZ (SPEC)	ANLZ (SPEC) (NS) (APC)	-1	RL LNG	7	5510	3	ones	2
TE2	36	Second residual	--	ANLZ (INVSF)	ANLZ (GAIN)	1	RL LNG	216	6062	3	--	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TFAC	37	Factors for Noise Shaping Analysis	--	BBN16I	ANLZ (NS)	1	RL LNG	6	6494	3	.684128 .468032 .320194 .219054 .149861 .102524	1
TANS	38	Noise shaping filter Coeffs	(see TCH)	ANLZ (NS)	ANLZ (APC)	1	RL LNG	6	5530	3	--	2
TGFAC	39	Quantizer Scale factors alternating with their reciprocals	TGFAC: GFAC(1), 1/GFAC(1), GFAC(2), 1/GFAC(2), GFAC(3), 1/GFAC(3)	ANLZ (GAIN)	ANLZ (APC)	1	RL LNG	6	6506	3	--	2
TIDG	40	Coded Delta Gains	--	ANLZ (GAIN)	ANLZ (GTHR)	1	FXD LNG	3	6518	3	--	2
TUO	41	Unquantized, normalized Residual (for debugging only)	--	ANLZ (APC)	SYNZ (SPECF)	1	RL LNG	216	1394	2	--	2
TQI	42	History for Noise-Shaping Filter	--	BBN16I ANLZ (APC)	ANLZ (APC)	1	RL LNG	6	6522	3	zeros	2
TVH	43	History for Spectral Filter	--	BBN16I ANLZ (APC)	ANLZ (APC)	1	RL LNG	6	6534	3	zeros	2

MAP-300 Buffers

Usage key: 1 = constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TRHØ	44	APC Pitch Filter History (Current frame)	←216→|←216→ ←TRHØ→ ←TRH→	ANLZ (APC)	ANLZ (APC)	1	RL LNG	(216)	6978	3	--	2
TRH	45	APC Pitch Filter History (last frame, current frame)	(see TRHØ)	BBN161 ANLZ (APC) (THIST)	ANLZ (APC)	1	RL LNG	432	6546	3	zeros	2
TSNKA (TIUA)	46	Coded transmission Data for ANLZA: TIUA, TIG, TIDG, TIM, TIC, TIK	--	ANLZ (APC) (GTHR)	PROPAR, PRORES	1	FXD LNG	230	41344	1	zeros	2
TSNKB (TIUB)	47	Coded transmission data for ANLZB: TIUB, TIG, ...	--	ANLZ (APC) (GTHR)	PROPAR, PRORES	1	FXD LNG	230	41574	1	zeros	2
TBTA	--	TBITS A Buffer	--	PROPAR, PRORES	TMINT	1	FXD LNG	522	41804	1	--	2
TBTB	--	TBITS B Buffer	--	PROPAR, PRORES	TMINT	1	FXD LNG	522	42326	1	--	2
TBTC	--	TBITS C Buffer	--	BBN161	TMINT	1	FXD LNG	522	42848	1	Bitstream equivalent of "Silence" (coded energy 0)	1

<u>MAP-300 Buffers</u>

Usage key: 1 - constant
2 = variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
TMDMA	--	TMODEM A Buffer	--	TMINT	MODEM Scroll Pgm	1	FXD LNG	522	43370	1	Bitstream equivalent of "silence" (coded energy - Ø) sync bit - Ø	2
TMDMB	--	TMODEM B Buffer	--	TMINT	MODEM Scroll Pgm	1	FXD LNG	522	43892	1	Bitstream equivalent of "silence" (coded energy Ø) sync bit - 1	2
RMDMA	--	RMODEM A Buffer	--	Modem Scroll Pgm	RMINT	1	FXD LNG	522	44414	1	--	2
RMDMB	--	RMODEM B Buffer	--	Modem Scroll Pgm	RMINT	1	FXD LNG	522	44936	1	--	2
RMDMC	--	RMODEM C Buffer	--	Modem Scroll Pgm	RMINT	1	FXD LNG	522	45458	1	--	2
RBTA	--	RBITS A Buffer	--	RMINT	DECODA/B, CORPAR	1	FXD LNG	522	45980	1	--	2
RBTB	--	RBITS B Buffer	--	RMINT	DECODA/B, CORPAR	1	FXD LNG	522	46502	1	--	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RSRA (RUHA)	48	Received decoded transmission data: RUHA, RGH, RDGH, RMH, RCH, RKH	--	DECODA/B, CORPAR	SYNZ (SPECF) (DEAL)	1	RL LNG	230	47024	1	zeros	2
RSRB (RUHB)	49	Received decoded transmission data: RUHB, RGH, RDGH, RMH, RCH, RKH	--	DECODA/B, CORPAR	SYNZ (SPECF) (DEAL)	1	RL LNG	230	47484	1	zeros	2
RDGH	50	Received Decoded $\sqrt{\text{Delta Gains}}$	--	SYNZ (DEAL)	SYNZ (SPECF)	1	RL LNG	3	7410	3	--	2
RCH	51	Received Decoded Pitch Predictor Coefficients	--	SYNZ (DEAL)	SYNZ (PITF)	1	RL LNG	3	7416	3	--	2
RKH	52	Received Decoded Reflection Coefficients	--	SYNZ (DEAL)	SYNZ (SPECF)	1	RL LNG	8	7422	3	zeros	2
RUH	53	Scaled Residual Samples	--	SYNZ (SPECF)	SYNZ (SPECF)	1	RL LNG	216	1826	2	--	2
RVHM	54	Spectral Synthesis Filter Memory	--	SYNZ (SPECF)	SYNZ (SPECF)	1	RL LNG	8	7438	3	zeros	2

MAP-300 Buffers

Usage key: 1 = constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RVH	55	Output of Spectral Synthesis Filter	--	SYNZ (SPECF)	SYNZ (PITF)	1	RL LNG	216	7454	3	--	2
RRHØ	56	Output of Pitch Synthesis Filter (current frame)	←216→ ←216→ ←RRHØ→ ←RRH→	SYNZ (PITF)	SYNZ (DEMP) (RHIST)	1	RL LNG	(216)	2690	2	--	2
RRH	57	Output of Pitch Synthesis Filter (last frame and current filter)	(see RRHØ)	SYNZ (RHIST) (PITF)	SYNZ (PITF)	1	RL LNG	432	2258	2	zeros	2
RSDEA (RRHDA)	58	Deemphasized synthetic speech	--	SYNZ (DEMP)	DAINT	1	RL SHRT	216	47944	1	--	2
RSDEB (RRHDB)	59	Deemphasized synthetic speech	--	SYNZ (DEMP)	DAINT	1	RL SHRT	216	48160	1	--	2
RSNFC	--	Synthetic "silence" data	--	BBN16I	DAINT	1	RL SHRT	216	49376	1	zeros	1
RDMPA	--	D/A Output to AOM	--	DAINT	AOM Scroll Program	1	RL SHRT	216	48592	1	zeros	2

MAP-300 Buffers

Usage key: 1 - constant
2 - variable

Name	BID	Description	Positional Characteristics	Written by:	Read by:	Sample Incr't	Sample Type	Number of Samples	Halfword Address	Bus #	Initial Values	Usage
RDABB	--	D/A Output to AOM	--	DAINT	AOM Scroll Program	1	RL SHRT	216	48808	1	zeros	2

APPENDIX C
MAP-300 SCALARS

The tables contained in this Appendix describe the characteristics of each user-defined real and integer scalar referenced in the MAP-300 speech coder implementation. The table entry labeled "SID" or "ISID" indicates the real or integer scalar identification number. The entries labeled "Written by:" and "Read by:" indicate subroutines, function lists, or program modules that so access each scalar. (The A/D, D/A, transmitter modem, and receiver modem interrupt service routines are specified in these tables by "ADINT", "DAINT", "TMINT", and "RMINT", respectively.)

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TALPH	50	Preemphasis filter coefficient	-	BBN16I	ANLZ (PREMP)	0.4	1
TPHST	51	Preemphasis filter history	-	ANLZ (PREMP)	ANLZ (PREMP)	0	2
TDCN	52	Negative of current frame's DC term	-	ANLZ (PITCH)	ANLZ (PITCH)	-	2
TFSZI	53	$-\frac{1}{\text{(Size of Pitch Extraction Buffer)}}$	-	BBN16I	ANLZ (PITCH)	$-\frac{1.}{265.}$	1
TMH	54	Pitch Lag (in samples)	-	ANLZ (PITCH)	ANLZ (INVPF) (APC)	-	2
TIM	55	Coded Pitch Lag (Integer in left hword)	-	ANLZ (PITCH)	ANLZ (GTBR)	-	2
T14	56	Constant 14	-	BBN16I	ANLZ (PITCH)	14.	1

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TKTHR	57	Threshold value for MEWL and MWLQ6	-	BBN16I	ANLZ (PITCH) (SPEC)	.9995	1
TIG	58	Coded Frame Energy (Integer in left hword)	-	ANLZ (GAIN)	ANLZ (GTHR)	-	2
TC2P	59	Pitch Predictor Coefficient for Single Tap Case	-	ANLZ (PITCH)	ANLZ (PITCH)	-	2
T72R	60	Constant: Reciprocal of 72	-	BBN16I	ANLZ (APC)	$\frac{1.}{72.}$	1
T3R	61	Constant: Reciprocal of 3	-	BBN16I	ANLZ (APC)	$\frac{1.}{3.}$	1
RGH	65	Received $\sqrt{\text{Frame Energy}}$	-	SYNZ (DEAL)	SYNZ (SPECF)	-	2
RMH	66	Received Pitch	-	SYNZ (DEAL)	SYNZ (PITF)	-	2

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RDCF0	67	Deemphasis filter coefficient SA0	Must be consecutive	BBN16I	SYNZ (DEMP)	0.	1
RDCF1	68	Deemphasis filter coefficient SA1	Must be consecutive	BBN16I	SYNZ (DEMP)	0.	1
RDCF2	69	Deemphasis filter coefficient SA2	Must be consecutive	BBN16I	SYNZ (DEMP)	- ALPHA = -0.4	1
RDCF3	70	Deemphasis filter coefficient SA3	Must be consecutive	BBN16I	SYNZ (DEMP)	0.	1
RDMY0	71	Deemphasis filter memory Y(-2)	Must be consecutive	SYNZ (DEMP)	SYNZ (DEMP)	0.	2
RDMY1	72	Deemphasis filter memory Y(-1)	Must be consecutive	SYNZ (DEMP)	SYNZ (DEMP)	0.	2
RDMY2	73	Deemphasis filter memory U(-2)	Must be consecutive	SYNZ (DEMP)	SYNZ (DEMP)	0.	2

MAP-300 REAL SCALARS

Usage key: 1 = constant
2 = variable

Name	SID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RDMY3	74	Deemphasis filter memory D(-1)	Must be consecutive	SYNZ (DEMP)	SYNZ (DEMP)	0.	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
TSRFA	50	TSRA Buffer Status Flag (0-> Empty)	-	ANLZ, ADINT	BBNLP, ADINT	0	2
TSRFB	51	TSRB Buffer Status Flag (0-> Empty)	-	ANLZ, ADINT	BBNLP, ADINT	0	2
TBTFA	52	TBTA Buffer Status Flag (0-> Empty)	-	ANLZ, TMINT	BBNLP, TMINT	0	2
TBTFB	53	TBTB Buffer Status Flag (0-> Empty)	-	ANLZ, TMINT	BBNLP, TMINT	0	2
TINC	54	MEWL output: Number of iterations not completed	Must be consecutive	ANLZ (PITCH)	-	-	2
TFENG	55	MEWL output: Fractional energy E(N)/R(0)	Must be consecutive	ANLZ (PITCH)	-	-	2
-	56	(Unused)	-	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RBTFA	57	RBTA Buffer Status Flag (0-> Empty)	-	SYNZ, RMINT	BBNLP, RMINT	0	2
RBTFB	58	RBTB Buffer Status Flag (0-> Empty)	-	SYNZ, RMINT	BBNLP, RMINT	0	2
RSNFA	59	RSNKA Buffer Status Flag (0-> Empty)	-	SYNZ, DAINT	BBNLP, DAINT	0	2
RSNFB	60	RSNKB Buffer Status Flag (0-> Empty)	-	SYNZ, DAINT	BBNLP, DAINT	0	2
RUN	61	System Run Flag (0-> Stop)	Must be consecutive	BBNRTS, BBNI6E	BBNRTS	0	2
-	62	(Left unused for MPITM)	Must be consecutive	-	-	-	-
-	63	(Unused)	-	-	-	-	-

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
ADPO	64	TADBA/B Buffer Pointer (0->B; -2->A)	-	ADINT	ADINT	0	2
TSRPO	65	TSRA/B Buffer Pointer (0->B; -2->A)	-	ADINT	ADINT	-2	2
TBTPO	66	TBTA/B Buffer Pointer (0->B; -2->A)	-	TMINT	TMINT	0	2
TMPO	67	TMDMA/B Buffer Pointer (0->B; -2->A)	-	TMINT	TMINT	0	2
RMPO	68	RMDMA/B/C Buffer Pointer (0=>C; -2=>B; -4->A)	-	RMINT	RMINT	0	2
RBTPO	69	RBTA/B Buffer Pointer (0->B; -2->A)	-	RMINT	RMINT	-2	2
RSNPO	70	RSNEA/B Buffer Pointer (0->B; -2->A)	-	DAINT	DAINT	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
DAPO	71	RDABA/B Buffer Pointer (0->B; -2->A)	-	DAINT	DAINT	0	2
TADDC	72	A/D Frame Discard Counter	-	ADINT	BBN16E	0	2
TMFFC	73	TMODEM Fake Frame Counter	-	TMINT	BBN16E	0	2
RMFDC	74	RMODEM Frame Discard Counter	-	RMINT	BBN16E	0	2
RSNNR	75	RSINK Data-Not-Ready Counter	-	DAINT	BBN16E	0	2
TFCTR	76	Transmitter Frame Counter	-	ADINT	BBN16E	0	2
RFCTR	77	Receiver Frame Counter	-	RMINT	BBN16E	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RLCTR	78	Receiver Lost-Sync Counter	-	RMINT	BBN16E	0	2
RONHK	79	Local Handset On-Hook State (0->Handset Off-Hook)	-	RMINT	ADINT, BBN16E	0	2
RSYNC	80	State of Receiver Sync 0->Lost Sync -1->Searching Sync +1->Found Sync	-	RMINT	RMINT BBN16E	0	2
RBOFO	81	Beginning-of-Frame Offset (Sync Bit Position in RMODEM Buffer)	Must be consecutive	RMINT	RMINT BBN16E	-	2
-	82	(Left unused for MFITM)	Must be consecutive				
ADCTR	83	A D (ADAM) Interrupt Counter	-	Device 23, line 1 interrupt service routine	BBN16E	0	2
TMCTR	84	TMODEM (IOS-2) Interrupt Counter	-	Device 16 line 1 interrupt service routine	BBN16E	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
RMCTR	85	RMODEM (IOS-2) Interrupt Counter	-	Device 16 line 2 interrupt service routine	BBN16E	0	2
DACTR	86	D/A (AOM) Interrupt Counter	-	Device 22, line 1 interrupt service routine	BBN16E	0	2
TADPK	87	A/D Peak Input Level	-	ADPEAK, BBN16E	BBN16E	0	2
-	88-122	(Unused)	-	-	-	-	-
RNCOR	123	Error-Correction switch (0->Error Correction)	Must be consecutive	BBN16E	CORPAR	0	2
-	124	(Left unused for MPITM)	Must be consecutive	-	-	-	-
ERRG	125	Set to non-zero to cause channel error simulation	Must be consecutive	BBN16E	RMINT	0	2

MAP-300 INTEGER SCALARS

Usage key: 1 = constant
2 = variable

Name	ISID	Description	Positional Characteristics	Written by:	Read by:	Initial Value	Usage
-	126	(Left unused for MPITM)	Must be consecutive	-	-	-	-
-	127	(Unused)	-	-	-	-	-

APPENDIX D
PROGRAM LISTINGS

This Appendix contains listings of the BBN-written (and modified) MAP-300 and PDP-11 (FORTRAN) programs referred to in Sections 2.4 and 3.

Page

[BBNO]<DCA16>BBN16M.MSU.2, 29-Dec-80 14:30:40, Ed: WOLF

T A B L E   O F   C O N T E N T S

```
                   (00001) ;[BBNDI<DCA16>BBN16M.MSU.2, 29-Dec-80 14:30:40, Ed: WOLF
                   (00003) *
                   (00004) * DEFINE "SAF" (SET ADDRESS FIELD) AND "LAF" (LOAD
                   (00005) * ADDRESS FIELD) INSTRUCTIONS TO ASSEMBLER.
                   (00006) * THESE INSTRUCTIONS ARE NEW CSPU INSTRUCTIONS
                   (00007) * AVAILABLE WITH "REV 10" CSPU MICROCODE.
                   (00008) * THE. SET/LOAD THE LOW ORDER 17 BITS OF THE
                   (00009) * REFERENCED FULL-WORD FROM/INTO THE INDICATED
                   (00010) * REGISTER.
                   (00011) *
                   (00012)        OPADD   SAF,(1 .LS. 14) + (29 .LS. 8) + $EF
                   (00013)        OPADD   LAF,(1 .LS. 14) + (29 .LS. 8) + $EE
                   (00014) *
                   (00015) *
                   (00016) * DEFINE SYMBOLS FOR ARRAY FUNCTION ASSEMBLIES
                   (00017) *
                   (00018) *
                   (00019) * FROM THE SNAP-II EXECUTIVE  ---  REL. 3.05  --- 5/22/79
                   (00020) *
                   (00021) *
         000008E8  (00022)        AFDT$ORG = $8E8
         00000245  (00023)        AP$ASSS = $245
         00000EFA  (00024)        AP$BNDR = $EFA
         00000F63  (00025)        AP$BNDR0 = $F63
         00000F2D  (00026)        AP$BNDR1 = $F2D
         0000024D  (00027)        AP$BSL = $24D
         00000248  (00028)        AP$CSSC = $248
         00000FB1  (00029)        AP$DONE = $FB1
         00000FBC  (00030)        AP$DONFR = $FBC
         0000000C  (00031)        AP$G0 = $C
         0000000D  (00032)        AP$G1 = $D
         00000011  (00033)        AP$LLOC = $11
         00000010  (00034)        AP$PBFF = $10
         0000000A  (00035)        AP$SAID = $A
         0000000E  (00036)        AP$SSCLR = $E
         00000009  (00037)        AP$SSS = $9
         00000004  (00038)        APS$BMO = $4
         0001FFC0  (00039)        APS$LA = $1FFC0
         0001FFC0  (00040)        APSLOC = $1FFC0
         0001FFC0  (00041)        APSHEM = $1FFC0
         0001FFC0  (00042)        APSR = $1FFC0
                   (00043) *
         00000604  (00044)        BCT$AD = $604
         00000686  (00045)        BCT$AT = $686
         00000582  (00046)        BCT$BA = $582
         00000000  (00047)        BIT$G0 = $0
         00000000  (00048)        BIT$RC = $0
         00000003  (00049)        BIT$WS = $3
                   (00050) *
         00001A06  (00051)        CK4DB$ = $1A06
         00000792  (00052)        CLR$G0G1 = $792
         00003190  (00053)        COS$ = $3190
```

```
00002IFC (00054)        CSPU$NOS = $21FC
         (00055) *
00000794 (00056)        DMY$ = $794
         (00057) *
00001AFA (00058)        ERROR$ = $1AFA
         (00059) *
00003110 (00060)        F11A$ = $3110
000007E8 (00061)        FDT$ = $7E8
00003040 (00062)        FEOS$ = $3040
0000079A (00063)        FFT$CBSZ = $79A
00000000 (00064)        FLG$CLR = $0
00000004 (00065)        FLG$G0 = $4
00000005 (00066)        FLG$G1 = $5
00000006 (00067)        FLG$G2 = $6
00000007 (00068)        FLG$G3 = $7
00000011 (00069)        FLG$RI = $11
00000020 (00070)        FLG$SET = $20
         (00071) *
00001B3C (00072)        GATHER$ = $1B3C
         (00073) *
00000001 (00074)        H$ = 1
         (00075) *
00000502 (00076)        ISVT$ = $502
         (00077) *
00001B7E (00078)        LOAD$AP = $1B7E
00001B8F (00079)        LOAD$AP1 = $1B8F
         (00080) *
00000006 (00081)        MSK$BMB = $6
0000FF00 (00082)        MSK$LBYT = $FF00
000000FF (00083)        MSK$RBYT = $00FF
00000000 (00084)        M$$ = 0
         (00085) *
0000001B (00086)        OQE = $1B
         (00087) *
000035EA (00088)        S1011$=$35EA
000007AE (00089)        SHFTL$R5 = $7AE
000023FA (00090)        SINCOS$T = $23FA
00003192 (00091)        SIN$ = $3192
00000382 (00092)        SVT$ = $382
000003CE (00093)        SVT$UN1 = $3CE
0001FFCE (00094)        SYS$FLGS = $1FFCE
         (00095) *
000007B4 (00096)        TEM$0 = $7B4
000021FE (00097)        TUE$ = $21FE
00000288 (00098)        TUE$PTR = $288
         (00099) *
00001C5A (00100)        VALBUF$ = $1C5A
00002870 (00101)        VSHA2$=$2870
         (00102) *
00000002 (00103)        W$ = $2
         (00104) *
0000192C (00105)        XFL$01 = $192C
         (00106) *
```

```
          0000078A (00107)         ZERO = $78A
                   (00108) *
                   (00109) *
                   (00110) *
          00000288 (00111) $L = TOE$PTR
                   (00112) *
    00288 001073C0 (00113)         ADDR   TOE$CUR(,1)            UPDATE TOP OF EXEC POINTER
                   (00114) *
                   (00115) *
          00000003 (00116)         $N = 3
                   (00117) *
                   (00118) *
                   (00119) *
```

```
                    (00120) *ARRAY FUNCTION DISPATCH TABLE PATCHES
                    (00121) *
                    (00122) *
                    (00123) *
          00000900  (00124)        #L = AFDT$ORG+3*W$*(132-128)    ;FCB 132  (VLTSY)
00900     001F5D02  (00125)        ADDR    VLTSY$(R7,1)
00902     001F5DC6  (00126)        ADDR    V3200$(R7,1)
00904     001021FC  (00127)        ADDR    CSPU$NOS(,1,0)
                    (00128) *
          00000906  (00129)        #L = AFDT$ORG+3*W$*(133-128)    ;FCB 133  (VKTOA)
00906     001F5E48  (00130)        ADDR    VKAU$(R7,1)
00908     001F5F76  (00131)        ADDR    VKASI(R7,1)
0090A     001021FC  (00132)        ADDR    CSPU$NOS(,1,0)
                    (00133) *
          0000090C  (00134)        #L = AFDT$ORG+3*W$*(134-128)    ;FCB 134  (PITCH)
0090C     001F5FE4  (00135)        ADDR    PITU$(R7,1)
0090E     001F5FA4  (00136)        ADDR    PITS$(R7,1)
00910     001021FC  (00137)        ADDR    CSPU$NOS(,1,0)
                    (00138) *
          00000912  (00139)        #L = AFDT$ORG+3*W$*(135-128)    ;FCB 135  (MWLQ6)
00912     001F6016  (00140)        ADDR    MWLF$APU(R7,1)
00914     001F60E8  (00141)        ADDR    MWLF$APS(R7,1)
00916     011061F2  (00142)        ADDR    MWLQ$SSM(,1,1)
                    (00143) *
          0000093C  (00144)        #L = AFDT$ORG+3*W$*(142-128)    ;FCB 142  (MEWL)
0093C     001F6206  (00145)        ADDR    MEWLS$(R7,1)
0093E     001F6316  (00146)        ADDR    MEWL$(R7,1)
00940     0010638D  (00147)        ADDR    SBM$MEWL(,1,0)
                    (00148) *
          0000096C  (00149)        #L = AFDT$ORG+3*W$*(150-128)    ;FCB 150  (STAB)
0096C     001F6444  (00150)        ADDR    STAB$(R7,1)
0096E     001F64D2  (00151)        ADDR    STABS$(R7,1)
00970     001021FC  (00152)        ADDR    CSPU$NOS(,1,0)
                    (00153) *
          000009D2  (00154)        #L = AFDT$ORG+3*W$*(167-128)    ;FCB 167  (INVPF)
009D2     001F651A  (00155)        ADDR    INVPU$(R7,1)
009D4     001F6562  (00156)        ADDR    INVPSI(R7,1)
009D6     001021FC  (00157)        ADDR    CSPU$NOS(,1,0)
                    (00158) *
          00000A2C  (00159)        #L = AFDT$ORG+3*W$*(182-128)    ;FCB 182  (ADPEAK)
00A2C     001F65D2  (00160)        ADDR    ADPKU(R7,1)
00A2E     001F65FE  (00161)        ADDR    ADPKA(R7,1)
00A30     001021FC  (00162)        ADDR    CSPU$NOS(,1,0)
                    (00163) *
          00000A32  (00164)        #L = AFDT$ORG+3*W$*(183-128)    ;FCB 183  (APUDMP)
00A32     001F6620  (00165)        ADDR    DUMPU(R7,1)
00A34     001F6660  (00166)        ADDR    DUMPS(R7,1)
00A36     001021FC  (00167)        ADDR    CSPU$NOS(,1,0)
                    (00168) *
          00000A5C  (00169)        #L = AFDT$ORG+3*W$*(190-128)    ;FCB 190  (HFC)
00A5C     001F6680  (00170)        ADDR    HFCU$(R7,1)
00A5E     001F6700  (00171)        ADDR    HFCS$(R7,1)
00A60     001021FC  (00172)        ADDR    CSPU$NOS(,1,0)
```

ARRAY FUNCTION DISPATCH TABLE PATCHES

```
                  (00173) *
      00000A80 (00174)          #L = AFDT$ORG+3*W$*(196-128)    ;FCB 196  (GAIN)
00A80 001F6764 (00175)          ADDR     GAIN$(R7,1)
00A82 001F6874 (00176)          ADDR     GAINS$(R7,1)
00A84 001021FC (00177)          ADDR     CSPU$NOS(,1,0)
                  (00178) *
      00000A86 (00179)          #L = AFDT$ORG+3*W$*(197-128)    ;FCB 197  (FEOFL)
00A86 001E6920 (00180)          ADDR     FEOSL$(R7,1)
00A88 001F69F0 (00181)          ADDR     F11AL$(R7,1)
00A8A 00106906 (00182)          ADDR     FEOF$SSH(,1,0)
                  (00183) *
      00000A8C (00184)          #L = AFDT$ORG+3*W$*(198-128)    ;FCB 198  (FEOIL)
00A8C 001E6920 (00185)          ADDR     FEOSL$(R7,1)
00A8E 001F69F0 (00186)          ADDR     F11AL$(R7,1)
00A90 00106902 (00187)          ADDR     FEOI$SSH(,1,0)
                  (00188) *
      00000A92 (00189)          #L = AFDT$ORG+3*W$*(199-128)    ;FCB 199  (APC)
00A92 001E6A72 (00190)          ADDR     APCU$(R7,1)
00A94 001F6BD0 (00191)          ADDR     APCS$(R7,1)
00A96 00106C90 (00192)          ADDR     SDM$APC(,1,0)
                  (00193) *
      00000A98 (00194)          #L = AFDT$ORG+3*W$*(200-128)    ;FCB 200  (GTHR)
00A98 001F6E32 (00195)          ADDR     GTHRU$(R7,1)
00A9A 001F6F42 (00196)          ADDR     GTHRS$(R7,1)
00A9C 001021FC (00197)          ADDR     CSPU$NOS(,1,0)
                  (00198) *
      00000A9E (00199)          #L = AFDT$ORG+3*W$*(201-128)    ;FCB 201  (THIST)
00A9E 001E6E9A (00200)          ADDR     THU$(R7,1)
00AA0 001F6FAA (00201)          ADDR     THS$(R7,1)
00AA2 00106EF4 (00202)          ADDR     SBM$TH(,1,0)
                  (00203) *
      00000AA4 (00204)          #L = AFDT$ORG+3*W$*(202-128)    ;FCB 202  (SCLRES)
00AA4 001F6FA2 (00205)          ADDR     SCL$(R7,1)
00AA6 001F6FDC (00206)          ADDR     SCLS$(R7,1)
00AA8 001021FC (00207)          ADDR     CSPU$NOS(,1,0)
                  (00208) *
      00000AAA (00209)          #L = AFDT$ORG+3*W$*(203-128)    ;FCB 203  (PITSYN)
00AAA 001E702E (00210)          ADDR     PITSU$(R7,1)
00AAC 001F7076 (00211)          ADDR     PITSSI(R7,1)
00AAE 001021FC (00212)          ADDR     CSPU$NOS(,1,0)
                  (00213) *
      00000AE0 (00214)          #L = AFDT$ORG+3*W$*(212-128)    ;FCB 212  (DEAL)
00AE0 001E70E8 (00215)          ADDR     DEALU$(R7,1)
00AE2 001E70F8 (00216)          ADDR     DEALS$(R7,1)
00AE4 001021FC (00217)          ADDR     CSPU$NOS(,1,0)
                  (00218) *
      00000AF6 (00219)          #L = AFDT$ORG+3*W$*(213-128)    ;FCB 213  (DCORZ)
00AF6 001E7150 (00220)          ADDR     DCOR$(R7,1)
00AF8 001F7262 (00221)          ADDR     ADC$(R7,1)
00AFA 001072FF (00222)          ADDR     SBDCORZ(,1,0)
                  (00223) *
```

```
                   (00224) *NON-ARRAY FUNCTION DISPATCH TABLE PATCHES
                   (00225) *
                   (00226) *
                   (00227) *
         000000BA  (00228)        $L = FDT$ + (WS * 105)        ;FCB 105  (MPIFF)
  000BA  0000739E  (00229)        ADDR    MPIFF$
                   (00230) *
         000000BC  (00231)        $L = FDT$ + (WS * 106)        ;FCB 106  (MPGSC)
  000BC  000073B0  (00232)        ADDR    MPGSC
                   (00233) *
```

```
                    (00234) "SYMBOL DEFINITIONS FOR BBN16 CODING/DECODING TABLE ADDRESSES
                    (00235) ;
                    (00236) ; TABLES RESIDE ON BUS1, STARTING AT $9800.
                    (00237) ; TABLE CONTENTS ARE DEFINED IN MODULE "BBN16T".
                    (00238) ;
          00009800  (00239) CTHC1=$9800            ;CODING TABLES
          00009810  (00240) CTHC2=CTHC1+ 8*W$
          00009800  (00241) CTLC3=CTHC1
          00009830  (00242) CTHK1=CTHC2+16*W$
          000098B0  (00243) CTHK2=CTHK1+64*W$
          000098F0  (00244) CTHK3=CTHK2+32*W$
          00009910  (00245) CTHK4=CTHK3+16*W$
          00009930  (00246) CTHK5=CTHK4+16*W$
          00009950  (00247) CTHK6=CTHK5+16*W$
          00009970  (00248) CTHG =CTHK6+16*W$
          000099F0  (00249) CTHDG=CTHG +64*W$
          000099F8  (00250) CTHU =CTHDG+ 4*W$
          00009A00  (00251) DVLC1=CTHU + 4*W$      ;DECODING TABLES
          00009A10  (00252) DVLC2=DVLC1+ 8*W$
          00009A00  (00253) DVLC3=DVLC1
          00009A30  (00254) DVLK1=DVLC2+16*W$
          00009AB0  (00255) DVLK2=DVLK1+64*W$
          00009AF0  (00256) DVLK3=DVLK2+32*W$
          00009B10  (00257) DVLK4=DVLK3+16*W$
          00009B30  (00258) DVLK5=DVLK4+16*W$
          00009B50  (00259) DVLK6=DVLK5+16*W$
          00009B70  (00260) DVLG =DVLK6+16*W$
          00009BF0  (00261) DVLDG=DVLG +64*W$
          00009BF8  (00262) DVLU =DVLDG+ 4*W$
          00009C00  (00263) DVLUF=DVLU + 4*W$
```

```
          'APU, APS, AND CSPU CODE FOR ADDED FUNCTIONS
          *
          * (FUNCTIONS ARE ORDERED BY FCD NUMBER, WITH
          *  ARRAY FUNCTIONS FIRST, THEN NON-ARRAY FUNCTIONS.)
          *
          *
          * SET START ADDRESS FOR ADDED FUNCTIONS.
          * REL 3.5 LEAVES EMPTY SPACE (ON BUS 1) AT:
          *          $3B00 - $4000
          *          $4B00 - $4A00
          *          $5D00 - $C000
          * PUT ADDED FUNCTIONS IN THIS SPACE.
          *
00005D00           $L = $5D00
          *
```

```
* APU3-VLTSY      VECTOR LATTICE SYNTHESIS FILTER
*
*       CODED BY LCOSELL, 3/79
*
*       BINDS TO APS3-V3200
*       PERFORMS LATTICE SYNTHESIS FILTER, USING REFL'N COEFFS
*
*       IMPLEMENTS THE FOLLOWING FORTRAN CODE:
*
*       DO 20 I=1,N
*
*       F(7)=W(I)-G(7)*K(8)
*       DO 10 J=6,0,-1
*       F(J)=F(J+1)-G(J)*K(J+1)
*10     G(J+1)=G(J)+F(J)*K(J+1)
*       G(0)=F(0)
*20     Y(I)=F(0)
*
*
* THERE ARE THREE INPUT VECTORS:
*       K (LENGTH 8) REFLECTION COEFFICIENTS (V BID)
*       G (LENGTH 8) FILTER MEMORY (U BID)
*       W (LENGTH N) INPUT RESIDUAL SAMPLES (W BID)
*THERE ARE TWO OUTPUT ARRAYS:
*       G AS BEFORE
*       Y OUTPUT SYNTHETIC SPEECH SAMPLES (LENGTH N) (Y BID)
*
*INPUT STREAM:
*       G(7)
*       G(6)
*       ...,
*       G(0)
*       K(8)
*       K(7)
*       ...,
*       K(1)
*       W(1)
*       W(2)
*       ...,
*       W(N).
*
*OUTPUT STREAM:
*       Y(1)
*       Y(2)
*       ...,
*       Y(N)
*       G(7)
*       G(6)
*       ...,
*       G(0).
*
*
*       REGISTER USAGE:
```

```
                        (00332) *LEFT          RIGHT
                        (00333) *       M0=G1         M0=G0
                        (00334) *       M1=G3         M1=G2
                        (00335) *       M2=G5         M2=G4
                        (00336) *       M3=G7         M3=G6
                        (00337) *NOTE: THESE ARE BEGINNING VALUES, ONLY.  AFFTER EACH G IS USED FOR THE FINAL TI
                        (00338) *THE REGISTER IS THEN USED FOR TEMP STORAGE FOR F(J), AND THEN FOR
                        (00339) *THE G'S FOR THE NEXT INPUT SAMPLE
                        (00340) *
                        (00341) *       M4=K2         M4=K1
                        (00342) *       M5=K4         M5=K3
                        (00343) *       M6=K6         M6=K5
                        (00344) *       M7=K8         M7=K7
                        (00345) *
                        (00346) *       A0=INPUT,TEMP A0=TEMP
                        (00347) *       A1=G1         A1=G0
                        (00348) *       A2=TEMP       A2=TEMP
                        (00349) *       A3=G3         A3=G2
                        (00350) *       A4=TEMP       A4=TEMP
                        (00351) *       A5=G5         A5=G4
                        (00352) *       A6=TEMP       A6=TEMP
                        (00353) *       A7=G7         A7=G6
                        (00354) *
                        (00355) *
                        (00356) *
                        (00357) *****************************************************************
                        (00358) *
                        (00359)         EVEN
     05D00 0000         (00360)         DATA    VLTSY$SA
     05D01 005E         (00361)         DATA    VLTSY$SZ
                        (00362) *
                        (00363) VLTSY$  BEGIN   APU(VLTSY)
               00000000 (00364)         $A=0
               00000000 (00365)         VLTSY$SA=$A
 A00 05D02 00CP0000 (00366) IN:     MOV(IQ,M3)\NOP  ;G7
 A01 05D04 00F70000 (00367)         MOV(IQA,A7)\NOP ;G7
 A02 05D06 000008CB (00368)         NOP\MOV(IQ,M3)  ;  ,G6
 A03 05D08 000008F7 (00369)         NOP\MOV(IQA,A7) ;  ,G6
 A04 05D0A 08CA0000 (00370)         MOV(IQ,M2)\NOP  ;G5
 A05 05D0C 08F50000 (00371)         MOV(IQA,A5)\NOP ;G5
 A06 05D0E 000008CA (00372)         NOP\MOV(IQ,M2)  ;  ,G4
 A07 05D10 000008F5 (00373)         NOP\MOV(IQA,A5) ;  ,G4
 A08 05D12 08C90000 (00374)         MOV(IQ,M1)\NOP  ;G3
 A09 05D14 08F30000 (00375)         MOV(IQA,A3)\NOP ;G3
 A0A 05D16 000008C9 (00376)         NOP\MOV(IQ,M1)  ;  ,G2
 A0B 05D18 000008F3 (00377)         NOP\MOV(IQA,A3) ;  ,G2
 A0C 05D1A 08C80000 (00378)         MOV(IQ,M0)\NOP  ;G1
 A0D 05D1C 08F10000 (00379)         MOV(IQA,A1)\NOP ;G1
 A0E 05D1E 000008C8 (00380)         NOP\MOV(IQ,M0)  ;  ,G0
 A0F 05D20 000008F1 (00381)         NOP\MOV(IQA,A1) ;  ,G0
 A10 05D22 08EF0000 (00382)         MOV(IQA,M7)\NOP ;K8
 A11 05D24 000008EF (00383)         NOP\MOV(IQA,M7) ;  ,K7
 A12 05D26 08EF0000 (00384)         MOV(IQA,M6)\NOP ;K6
```

```
A13 05D28 00300AEE (00385)        NOP\MOV(IQA,M6) ;  ,F5
A14 05D2A 85E805E8 (00386)        MUL(M3,M7)\MUL(M3,M7)    ;G7*K8,G6*K7
A15 05D2C 08ED0000 (00387)        MOV(IQA,M5)\NOP ;K4
A16 05D2E 00000AED (00388)        NOP\MOV(IQA,M5) ;  ,K3
A17 05D30 0AEC0000 (00389)        MOV(IQA,M4)\NOP ;K2
A18 05D32 00000AEC (00390)        NOP\MOV(IQA,M4) ;  ,K1
A19 05D34 0AF00000 (00391)        MOV(IQA,A0)\NOP ;F8=INPUT
A1A 05D36 08860000 (00392)        MOV(P,A6)\NOP   ;G7*K8
A1B 05D38 4E000000 (00393)        SUB(A0,A6)\NOP  ;F8-G7*K8>F7
A1C 05D3A 85408556 (00394)        MUL(M2,M6)\MOV(A6),MUL(M2,M6)    ;G5*K6,G4*K5
A1D 05D3C 08980000 (00395) LOOP:  MOV(R,EXO)\NOP  ;F7
A1E 05D3E 00000850 (00396)        NOP\MOV(EXI,A0) ;  ,F7
A1F 05D40 00004F00 (00397)        NOP\SUB(A0,A6)  ;  ,F7-G6*K7>F6
A20 05D42 00000A80 (00398)        NOP\MOV(R,M3)   ;  ,F6
A21 05D44 840485F4 (00399)        MOV(A4),MUL(M1,M5)\MOV(A4),MUL(M3,M7)   ;(G5*K6)G3*K4;(G4K5)F6K7
A22 05D46 00000A98 (00400)        NOP\MOV(R,EXO)  ;  ,F6
A23 05D48 08560000 (00401)        MOV(EXI,A6)\NOP ;F6
A24 05D4A 4CC00000 (00402)        SUB(A6,A4)\NOP  ;F6-G5K6>F5
A25 05D4C 08980000 (00403)        MOV(R,EXO)\NOP  ;F5
A26 05D4E 08A00852 (00404)        MOV(R,M2)\MOV(EXI,A2)     ;F5,F5
A27 05D50 85528486 (00405)        MOV(A2),MUL(M2,M6)\MOV(A6),MUL(M1,M5)   ;(G3K4)F5K6,(F6K7)G2K3
A28 05D52 00004C40 (00406)        NOP\SUB(A2,A4)  ;  ,F5-G4K5>F4
A29 05D54 00000898 (00407)        NOP\MOV(R,EXO)  ;  ,F4
A2A 05D56 08540000 (00408)        MOV(EXI,A4)\NOP ;F4
A2B 05D58 4A8046EA (00409)        SUB(A4,A2)\MOV(M2),ADD(A7,A6)    ;F4-G3K4>F3,(F4)G6+F6K7>G7
A2C 05D5A 84168552 (00410)        MOV(A6),MUL(M0,M4)\MOV(A2),MUL(M2,M6)   ;(F5K6)G1K2,(G2K3)F4K5
A2D 05D5C 08980898 (00411)        MOV(R,EXO)\MOV(R,EXO)     ;F3,G7
A2E 05D5E 08400854 (00412)        MOV(EXI,M3)\MOV(EXI,A4) ;G7(NEW),F3
A2F 05D60 46A94A80 (00413)        MOV(M1),ADD(A5,A6)\SUB(A4,A2)    ;(F3)G5+F5K6>G6,F3-G2K3>F2
A30 05D62 08570000 (00414)        MOV(EXI,A7)\NOP ;G7(NEW)
A31 05D64 84808414 (00415)        MOV(A0),MUL(M1,M5)\MOV(A4),MUL(M0,M4)   ;(G1K2)F3K4,(F4K5)G0K1
A32 05D66 00000898 (00416)        NOP\MOV(R,EXO)  ;  ,F2
A33 05D68 08520000 (00417)        MOV(EXI,A2)\NOP ;F2
A34 05D6A 405844A9 (00418)        MOV(EXO),SUB(A2,A0)\MOV(M1),ADD(A5,A4)  ;(G6)F2-G1K2>F1,(F2)G4+
                   (00419) *                                            ;F4K5>G5
A35 05D6C 00000857 (00420)        NOP\MOV(EXI,A7) ;  ,G6(NEW)
A36 05D6E 00000840 (00421)        NOP\MOV(EXI,M3) ;  ,G6(NEW)
A37 05D70 08980898 (00422)        MOV(R,EXO)\MOV(R,EXO)     ;F1,G5
A38 05D72 08900852 (00423)        MOV(R,M0)\MOV(EXI,A2)     ;F1,F1
A39 05D74 84148400 (00424)        MOV(A4),MUL(M0,M4)\MOV(A0),MUL(M1,M5)   ;(F3K4)F1K2,(G0K1)F2K3
A3A 05D76 440F4840 (00425)        ADD(A3,A4)\SUB(A2,A0)     ;G3+F3K4>G4,F1-G0K1>F0
A3B 05D78 08550000 (00426)        MOV(EXI,A5)\NOP ;G5(NEW)
A3C 05D7A 08410000 (00427)        MOV(EXI,M2)\NOP ;G5(NEW)
A3D 05D7C 08980888 (00428)        MOV(R,EXO)\MOV(R,M0)      ;G4,F0=G0(NEW)
A3E 05D7E 08078412 (00429)        MOV(P,A2)\MOV(A2),MUL(M0,M4)      ;F1K2,F0K1
A3F 05D80 42204278 (00430)        ADD(A1,A2)\MOV(EXO),ADD(A3,A2)   ;G1+F1K2>G2,(F0)G2+F2K3>G3
A40 05D82 085C0344 (00431)        MOV(EXI,OQ)\MOV(EXI,M2) ;F0 OUT,G4(NEW)
A41 05D84 08510855 (00432)        MOV(EXI,A1)\MOV(EXI,A5) ;F0=G0,G4(NEW)
A42 05D86 08980000 (00433)        MOV(R,EXO)\NOP  ;G2
A43 05D88 9110004F (00434)        JUMPS(LDONE,FI) ;END IF INPUT USED UP
A44 05D8A 0AF00000 (00435)        MOV(IQA,A0)\NOP ;F8(NEW)=INPUT
A45 05D8C 85E085F0 (00436)        MUL(M3,M7)\MOV(A0),MUL(M3,M7)    ;G7K8,(F0K1)G6K7
A46 05D8E 02204030 (00437)        R(A1)\MOV(EXO),ADD(A1,A0)        ;F0=G0,(G3)G0+F0K1>G1
```

```
A47 05D90 08490849 (00438)         MOV(EXI,M1)\MOV(EXI,M1) ;G3(NEW),G2(NEW)
A48 05D92 08530853 (00439)         MOV(EXI,A3)\MOV(EXI,A3) ;G3(NEW),G2(NEW)
A49 05D94 08900890 (00440)         MOV(R,EXO)\MOV(R,EXO)   ;G0=F0,G1
A4A 05D96 08400000 (00441)         MOV(EXI,M0)\NOP ;G1(NEW)
A4B 05D98 08510851 (00442)         MOV(EXI,A1)\MOV(EXI,A1) ;G1(NEW),G0(NEW)
A4C 05D9A 05560556 (00443)         MOV(A6),MUL(M2,M6)\MOV(A6),MUL(M2,M6)   ;(G7K8)G5K6,(G6K7)G4K5
A4D 05D9C 4E000000 (00444)         SUB(A0,A6)\NOP  ;F8-G7K8>F7
A4E 05D9E 1000001D (00445)         JUMP(LOOP)
                   (00446) *
                   (00447) *       FINISH LAST PART OF LAST LOOP, AND OUTPUT G'S
A4F 05DA0 00000080 (00448) LDONE:  NOP\MOV(P,A0)   ;  ,F0K1
A50 05DA2 02204038 (00449)         R(A1)\MOV(EXO),ADD(A1,A0)       ;F0=G0,(G3)G0+F0K1
A51 05DA4 08530853 (00450)         MOV(EXI,A3)\MOV(EXI,A3) ;G3(NEW),G2(NEW)
A52 05DA6 02F802F8 (00451)         MOV(EXO),R(A7)\MOV(EXO),R(A7)   ;(G0)G7,(G1)G6
A53 05DA8 089C0000 (00452)         MOV(R,OQ)\NOP   ;G7 OUT
A54 05DAA 02A002BC (00453)         R(A5)\MOV(OQ),R(A5)     ;G5,(G6 OUT),G4
A55 05DAC 089C0000 (00454)         MOV(R,OQ)\NOP   ;G5 OUT
A56 05DAE 0260027C (00455)         R(A3)\MOV(OQ),R(A3)     ;G3,(G4 OUT)G2
A57 05DB0 089C0000 (00456)         MOV(R,OQ)\NOP   ;G3 OUT
A58 05DB2 0000089C (00457)         NOP\MOV(R,OQ)   ;  ,G2 OUT
A59 05DB4 085C0000 (00458)         MOV(EXI,OQ)\NOP ;G1 OUT
A5A 05DB6 0000085C (00459)         NOP\MOV(EXI,OQ) ;  ,G0 OUT
A5B 05DB8 20320000 (00460)         CLEAR(RA)\NOP    ;HALT
A5C 05DBA 00000000 (00461)         NOP
A5D 05DBC 10000000 (00462)         JUMP(0)
                   (00463) *
          0000005E (00464)         VLTSY$SZ=#A-VLTSY$SA
    05DBE          (00465)         END     VLTSY$SZ
                   (00466)         EVEN
                   (00467) *
```

```
                        * APS3-V3200     APS PROGRAM FOR VLTSY
                        *
                        *        MAP-300 APS PROGRAM FOR VLTSY(Y,U,V,W)
                        *        CODED BY KFIELD 3/79
                        *
                        *        INPUT STREAM: U(7),...,U(0),V(7),...,V(0),W(0),...,W(WBS-1),[FI]
                        *
                        *        OUTPUT STREAM: Y(0),...,Y(WBS-1),U(7),...,U(0),[EO]
                        *
                        *
                        *        HEADER BLOCK
                        *
                                 EVEN
    05DBE 00005E2A               ADDR     V3200$I          ;PTR TO CONSTR. INSTR. BLK.
    05DC0 00000000               ADDR     0                ;PTR TO SCALAR BLOCK (NO SCALARS)
    05DC2 0000                   DATA     0                ;NUMBER OF SCALARS
    05DC3 0080                   DATA     V3200$Z          ;MODULE SIZE
    05DC4 00005E22               ADDR     V3200$A          ;PTR TO CHAIN ANCHOR
                                 EVEN
                        *
                        *
                        V3200$   BEGIN    APS(V3200)
                        *
                        *
                        *        INPUT PROGRAM
                        *
                        *
A00 05DC6 00201D40               JSN(V3200$6,P2)           ;SET OUTPUT PC
A01 05DC8 02300030               SET(RO)                   ;TURN ON OUTPUT GENERATION
                        *
                        *        INPUT PGM REGISTER USAGE:
                        *                BR0: VECTOR ELEMENT ADDRESSES
                        *                BR1: BUFFER SIZES
                        *                BR2: BUFFER SPACINGS
                        *
                        *
                        *        GENERATE "U" ADDRESSES
                        *
A02 05DCA 04401000               LOAD(BR0,[1])             ;BR0 <= VECTOR U BASE ADDR
A03 05DCC 06500000               LOAD(BR1,M$$)             ;DUMMY LOAD
A04 05DCE 08600000               LOAD(BR2,M$$)             ;BR2 <= VECTOR U SPACING
A05 05DD0 0A290031               SUBL(BR2,1)               ;BR2 <= SPACING-1
                        *
A06 05DD2 0C00003F      #1       ADDL(BR0,7)               ;GEN ADDR OF U(7)
A07 05DD4 0F2906B1               SUBL(BR2,1),JUMPP(#1)     ;ADD (7 * SPACING)
                        *
A08 05DD6 100A900C               ADD(BR0,[9])              ;BR0 <= ADDR(U7))+SPACING
                        *
A09 05DD8 12500007               LOAD(BR1,7)               ;BR1 <= 7
A0A 05DDA 14029004      #2       SUB(BR0,[9],TF)           ;GEN ADDR (U(I))
A0B 05DDC 16190AB1               SUBL(BR1,1),JUMPP(#2)     ;LOOP 8 TIMES
                        *
                        *
```

```
PAGE    14:  [RHMD]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
               APS3-V3200       APS PROGRAM FOR VLTSY

                          (00521) *        GENERATE 'V' ADDRESSES
                          (00522) *
  A0C 05DDE 18402004 (00523)          LOAD(BR0,[2])             ;BR0 <= VECTOR V BASE ADDR
  A0D 05DF0 1A500000 (00524)          LOAD(BR1,M$$)             ;DUMMY LOAD
  A0E 05DF2 1C600000 (00525)          LOAD(BR2,M$$)             ;BR2 <= VECTOR V SPACING
  A0F 05DF4 1E200031 (00526)          SUBL(BR2,1)               ;BR2 <= SPACING-1
                          (00527) *
  A10 05DF6 2009003F (00528) #3       ADDL(BR0,7)               ;GEN ADDR OF V(7)
  A11 05DF8 22291081 (00529)          SUBL(BR2,1),JUMPP(#3)     ;ADD (7 * SPACING)
                          (00530) *
  A12 05DEA 240AA00C (00531)          ADD(BR0,[10])             ;BR0 <= ADDR(V(7)) + SPACING
                          (00532) *
  A13 05DEC 26500007 (00533)          LOAD(BR1,7)               ;BR1 <= 7
  A14 05DFE 2882A004 (00534) #4       SUB(BR0,[10],TF)          ;GEN ADDR(V(I))
  A15 05DF0 2A191481 (00535)          SUBL(BR1,1),JUMPP(#4)     ;LOOP 8 TIMES
                          (00536) *
                          (00537) *
                          (00538) *        GENERATE 'W' ADDRESSES
                          (00539) *
  A16 05DF2 2C403004 (00540)          LOAD(BR0,[3])             ;BR0 <= VECTOR W BASE ADDR
  A17 05DF4 2E500000 (00541)          LOAD(BR1,M$$)             ;BR1 <= WBS-1
  A18 05DF6 30020000 (00542)          SUB(BR0,M$$)              ;BR2 <= W BASE MINUS SPACING
                          (00543) *
  A19 05DF8 328AB006 (00544) #5       ADD(BR0,[11],TF)          ;GEN ADDR(W(I))
  A1A 05DFA 341919B1 (00545)          SUBL(BR1,1),JUMPP(#5)     ;LOOP WBS TIMES
                          (00546) *
                          (00547) *        WHEN DONE GENERATING INPUT ADDRESSES, CLEAR RI
                          (00548) *
  A1B 05DFC 36200031 (00549)          CLEAR(RI)                 ;HALT INPUT
  A1C 05DFE 38000020 (00550)          NOP(0)
                          (00551) *
                          (00552) *
                          (00553) *        OUTPUT PROGRAM
                          (00554) *
                          (00555) *
                          (00556) *        OUTPUT PGM REG USAGE:
                          (00557) *                BW0: VECTOR ELEMENT ADDRESSES
                          (00558) *                BW1: BUFFER SIZES
                          (00559) *                BW2: BUFFER SPACINGS
                          (00560) *                BW3: SCRATCH REGISTER
                          (00561) *
  A1D 05E00 3A300032 (00562) V3200$6 SET(RA)                  ;TURN ON APU
                          (00563) *
                          (00564) *        GEN 'Y' ADDRESSES
                          (00565) *
  A1E 05E02 3C40000A (00566)          LOAD(BW0,[0])             ;BW0 <= Y BASE ADDR
  A1F 05E04 3E500000 (00567)          LOAD(BW1,M$$)             ;DUMMY LOAD
  A20 05E06 40020000 (00568)          SUB(BW0,M$$)              ;BW0 <= Y BASE MINUS SPACING
                          (00569) *
                          (00570) *        GET WBS-1  (GENERATE WBS Y'S)
                          (00571) *
  A21 05E08 42703006 (00572)          LOAD(BW3,[3])             ;SCRATCH <= W BASE
  A22 05E0A 44500000 (00573)          LOAD(BW1,M$$)             ;BW1 <= WBS-1
```

```
 A23 05E0C 46700000 (00574)          LOAD(BW3,H$$)          ;DUMMY LOAD
                    (00575) *
 A24 05E0E 488A0E06 (00576) #7       ADD(BW0,[8],TF)        ;GEN ADDR(Y(I))
 A25 05E10 4A1124B1 (00577)          SUBL(BW1,1),JUMPP(#7)  ;LOOP NBS TIMES
                    (00578) *
                    (00579) *        GENERATE 'U' ADDRESSES
                    (00580) *
 A26 05E12 4C401004 (00581)          LOAD(BW0,[1])          ;BW0 <= VECTOR U BASE ADDRESS
 A27 05E14 4F500060 (00582)          LOAD(BW1,H$$)          ;DUMMY LOAD
 A28 05E16 50600080 (00583)          LOAD(BW2,H$$)          ;BW2 <= VECTOR U SPACING
 A29 05E18 52210031 (00584)          SUBL(BW2,1)            ;BW2 <= SPACING-1
                    (00585) *
 A2A 05E1A 5401003F (00586) #8       ADDL(BW0,7)            ;GEN ADDR OF U(7)
 A2B 05E1C 56212AB1 (00587)          SUBL(BW2,1),JUMPP(#8)  ;ADD (7*SPACING)
                    (00588) *
 A2C 05E1E 580A900C (00589)          ADD(BW0,[9])           ;BW0 <= ADDR(U(7)) + SPACING
                    (00590) *
 A2D 05E20 5A500087 (00591)          LOAD(BW1,7)            ;BW1 <= 7
 A2E 05E22 5CB29004 (00592) #9       SUB(BW0,[9],TF)        ;GEN ADDR(U(I))
 A2F 05E24 5E112EB1 (00593)          SUBL(BW1,1),JUMPP(#9)  ;LOOP 8 TIMES
                    (00594) *
                    (00595) *
                    (00596) *
 A30 05E26 60200030 (00597)          CLEAR(RO)              ;HALT OUTPUT
 A31 05E28 62000020 (00598)          NOP(0)
                    (00599) *
           00005E22 (00600) V3200$A=#C                      ;ASSIGN VALUE TO CHAIN ANCHOR
     05E2A          (00601)          END      #A-1          ;END OF MODULE
                    (00602) *
                    (00603) *        CONSTR. INSTR. BLOCK
                    (00604) *
     05E2A 00000000 (00605) V3200$T DATA    14F'0.0'
     ...
           00000000 (00606) V3200$Z=#L-V3200$               ;MODULE SIZE
                    (00607) *
```

```
PAGE    17:  [BBND]<BCA16>BBN16M.MSD.2, 29-Dec-80 14:30:40, Ed: WOLF
               APU3-VKTOA(Y,V)  CONVERT REFL'N (K) TO LPC (A) COEFFICIENTS    .

  A05 05E52 00E000E0 (00661)          MOV(IQA,M0)                ;INPUT K(M)
                     (00662) ;INNER LOOP
  A06 05E54 00F000EC (00663) 02:      MOV(IQA,A0) \ MOV(IQA,M4)        ;A[M-1](M-L)
  A07 05E56 00EC00F0 (00664)          MOV(IQA,M4) \ MOV(IQA,A0)        ;A[M-1](L)
  A08 05E58 04000400 (00665)          MUL(M0,M4)                 ;K(M)*A[M-1](L) \ K(M)*A[M-1](M-L)
  A09 05E5A 00810081 (00666)          MOV(P,A1)
  A0A 05E5C 41004100 (00667)          ADD(A0,A1)                 ;A[M-1](M-L) + K(M)*A[M-1](L) \
                     (00668) ;                                   ;A[M-1](L) + K(M)*A[M-1](M-L)
  A0B 05E5E 009C009C (00669)          MOV(R,OQ)                  ;A[M](M-L) \ A[M](L)
  A0C 05E60 90160006 (00670)          JUMPC(02,FWI)              ;JUMP IF NOT DONE THIS ITERATION
  A0D 05E62 9010000D (00671) 03:      JUMPC(03,OQE)              ;WAIT FOR THIS ITERATION'S RESULT TO BE
  A0E 05E64 00000000 (00672)          NOP                        ; BE WRITTEN BEFORE LETTING APS PROCEED
  A0F 05E66 20372037 (00673)          CLEAR(WI)                  ; TO READ OPERANDS FOR NEXT ITERATION
  A10 05E68 901C0004 (00674)          JUMPC(01,EO)               ;JUMP IF MORE ITERATIONS YET TO DO
  A11 05E6A 20322032 (00675)          CLEAR(RA)                  ;HALT
  A12 05E6C 10000000 (00676)          JUMP(0)
            00000013 (00677) VKAU$SZ=$A-VKAU$SA
      05E6E          (00678)          END
```

```
                    ' APU3-VKTOA(V,U)  CONVERT REFL'N (K) TO LPC (A) COEFFICIENTS
                    ;
                    ;  J. J. WOLF  6/30/80
                    ;  BINDS TO APS3-VKTOA
                    ;
                    ;THIS VKTOA IS DIFFERENT FROM THAT IN BBN300.MSO (9.6 KB/S CODER)
                    ;  IN THAT (1) BOTH INPUT AND OUTPUT BUFFERS ARE FLT PT AND (2) THIS
                    ;  IS WRITTEN TO HANDLE A GENERAL COEFFICIENT VECTOR (ANY LENGTH).
                    ;
                    ;  BUFFER V: OUTPUT, LPC COEFFS: A(1), A(2), ..., A(N).  A(0), WHICH IS
                    ;       1.0, IS NOT OUTPUT.
                    ;  BUFFER U: INPUT, REFLECTION COEFFS, K(1), K(2), ..., K(N)
                    ;       NOTE: BUFFERS V AND U MUST BE COMPACT 32-BIT FLT PT.
                    ;
                    ;THE LPC COEFFICIENTS ARE COMPUTED FROM THE RECURSION:
                    ;  (1)  A[M](M) = K(M)
                    ;  (2)  A[M](L) = A[M-1](L) + K(M)*A[M-1](M-L)
                    ;                 M=1,2,...,N
                    ;                 L=1,2,...,M-1
                    ;  WHERE [.] INDICATES ITERATION AND (.) INDICATES INDEX.
                    ;IN ORDER THAT THE COMPUTATION BE DONE IN PLACE, WE DO EQ. (2) IN
                    ;  "SYMMETRIC" PAIRS:
                    ;  (2A) A[M](M-L) = A[M-1](M-L) + K(M)*A[M-1](L)
                    ;  (2B) A[M](L)   = A[M-1](L)   + K(M)*A[M-1](M-L)
                    ;                 L=1,2,...,(M-1)/2
                    ;
                    ;APU REGISTER USAGE:
                    ;  A(0):          A[M-1](M-L)      A[M-1](L)
                    ;  A(1):          K*A              K*A
                    ;  M(0):          K(M)             K(M)
                    ;  M(4):          A[M-1](L)        A[M-1](M-L)
                    ;
                    ;INPUT, OUTPUT STREAMS:
                    ;       M=1       M=2  L=1          M=3  L=1
                    ;IQ=  K(1)[WI], K(2),A(1),A(1)[WI], K(3),A(2),A(1)[WI],
                    ;OQ=  A(1),     A(2),A(1),A(1),     A(3),A(2),A(1),
                    ;       M=4  L=1        L=2
                    ;       K(4),A(3),A(1),A(2),A(2)[WI],...
                    ;       A(4),A(3),A(1),A(2),A(2),     ...[EO]
                    ;THE NUMBER OF ITERATIONS IS UBS.
                    ;
                            EVEN
    05E46 0000                  DATA VKAU$SA            ;START ADDRESS
    05E47 0013                  DATA VKAU$SZ            ;SIZE
                    ;
                    VKAU$:  BEGIN APU(VKAU)
                    VKAU$SA:
A00 05E48 00FC0000              MOV(IQA,OQ) \ NOP       ;A[1](1)=K(1).  END OF M=1 ITERATION.
A01 05E4A 90100001  010:        JUMPC(010,UQE)          ;WAIT FOR M=1 RESULT TO BE WRITTEN
A02 05E4C 00000000              NOP
A03 05E4E 28372037              CLEAR(WI)               ;THEN LET APS PROCEED
                    ;ITERATION LOOP
A04 05E50 00DC0000  01:         MOV(IQ,OQ) \ NOP        ;A[M](M)=K(M)
```

```
                             " APS3-VKTOA(V,U)    APS PROGRAM FOR VKTOA
                             ;  J. WOLF, 7/2/80
                             ;
                             ;HEADER BLOCK
                             ;
                                      EVEN
    05E6E 00005ED0                    ADDR VKAS$I            ;PTR TO CONSTR. INSTR. BLOCK
    05E70 00000000                    ADDR 0                 ;ADDR TO SCALAR BLOCK
    05E72 0000                        DATA 0                 ;NUMBER OF SCALARS
    05E73 006C                        DATA VKAS$Z            ;MODULE SIZE
    05E74 00005EBC                    ADDR VKAS$A            ;PTR TO CHAIN ANCHOR
                                      EVEN
                             ;
                             ;INPUT PROGRAM
                             ;  BR0: COUNTS ITERATIONS FROM 2 TO UBS
                             ;  BR1: K(M), A(X)
                             ;  BR2: MI=2*(M-1)
                             ;  BR3: M' COUNTS FROM 2*(M-1) BY 2 TO 0 OR -2
                             ;
                             VKASI:   BEGIN APS(VKAS)
A00 05E76 000D1000                    LOAD(BR1,C13,TF)                ;UBA, IQ: K(1) (M=1 ITERATION)
A01 05E78 02400000                    LOAD(BR0,M$$)                   ;UBS-1
A02 05E7A 04700000                    LOAD(BR3,M$$)                   ;(UNUSED) U-INCR
A03 05E7C 06090031                    SUBL(BR0,1)                     ;UBS-2 = ITERATION CTR
A04 05E7E 00010011                    MOVB(BW0,BR0)                   ;SAME CTR USED IN OUTPUT PGM
A05 05E80 0A201D40                    JSM(VKASO,P2)                   ;SET AND START OUTPUT PGM
A06 05E82 0C300030                    SET(RO)
A07 05E84 0E600000                    LOAD(BR2,0)                     ;INIT MI=0
A08 05E86 10300037                    SET(WI)                ;WAIT FOR M=1 ITERATION TO FINISH
A09 05E88 12000020                    NOP(0)
                             ;ITERATION LOOP.  WE BEGIN WITH THE M=2 ITERATION.
A0A 05E8A 1429003A           04:      ADDL(BR2,2)            ;UPDATE MI
A0B 05E8C 16501F16                    LOAD(BR1,C13)          ;UBA, K(1)
A0C 05E8E 18700000                    LOAD(BR3,M$$)          ;(UNUSED)
A0D 05E90 1A700000                    LOAD(BR3,M$$)          ;(UNUSED)
A0E 05E92 1C99002D                    ADDB(BR1,BR2,TF)       ;IQ: K(M)
A0F 05E94 1E500000                    LOAD(BR1,C03)          ;VBA, A(1)
A10 05E96 20700000                    LOAD(BR3,M$$)          ;(UNUSED) VBS-1
A11 05E98 22120000                    SUB(BR1,M$$)           ;"A(0)"
A12 05E9A 24391455                    MOVB(BR3,BR2),JUMP(VK6)  ;M'=2*(M-1) AND ENTER LOOP
                             ;INNER LOOP: L=1,2,...,(M-1)/2
A13 05E9C 2639003A           05:      ADDL(BR3,W$)           ;ADD W$ SUBTRACTED OUT AT LOOP TEST
A14 05E9E 2899002F           VK6:     ADDB(BR1,BR3,TF)       ;IQ: A(M-L)
A15 05EA0 2A390032                    SUBL(BR3,W$)
                             ;THE NEXT 3 INSTRUCTIONS ARE UNINTERRUPTIBLE, ENSURING THAT ON THE FINAL
                             ;  LOOP OF THE ITERATION, WI GETS SET 2 APU-CYCLES AFTER THE LAST READ
                             ;  ADDRESS IS GENERATED, SO IT'S SURE TO BE SET BEFORE THE APU CAN TEST
                             ;  IT.
A16 05EA2 2C991767                    SUBB(BR1,BR3,TF),JUMP(0A+1)  ;IQ: A(L)
A17 05EA4 2E3913B4                    SUBL(BR3,2*W$),JUMPP(05)  ;TEST M' TO SEE IF END OF ITERATION
A18 05EA6 30300037                    SET(WI)                ;END OF ITERATION: WAIT TIL APU FINDS
A19 05EA8 32000020                    NOP(0)                 ; DQ EMPTY BEFORE GOING ON TO NEXT
A1A 05EAA 34090A81                    SUBL(BR0,1),JUMPP(04)  ;JUMP IF MORE ITERATIONS TO DO
```

```
 A1B 05EAC 36200031 (00732)          CLEAR(RI)                ;OTHERWISE, IT'S TIME TO HALT
 A1C 05EAE 30000020 (00733)          NOP(0)
                    (00734) ;
                    (00735) ;OUTPUT PROGRAM
                    (00736) ;  BW0: ITERATION COUNTER, SAME AS INPUT PGM
                    (00737) ;  BW1: A(X)
                    (00738) ;  BW2: M1=2*(M-1)
                    (00739) ;  BW3:  M' COUNTS FROM 2*(M-1) BY 2 TO 0 OR -2
                    (00740) ;
 A1D 05EB0 3A300032 (00741) VKASO:   SET(RA)                  ;START APU
 A1E 05EB2 3CD0001E (00742)          LOAD(BW1,C0],TF)         ;YBA, OQ: A(1), END OF N=1 ITERATION
 A1F 05EB4 3E700000 (00743)          LOAD(BW3,M$$)            ;(UNUSED) YBS-1
 A20 05EB6 40700000 (00744)          LOAD(BW3,M$$)            ;(UNUSED) YINC
 A21 05EB8 42600000 (00745)          LOAD(BW2,0)              ;INIT M1 TO 0
                    (00746) ;ITERATION LOOP.  WE BEGIN WITH N=2 ITERATION
 A22 05EBA 4421003A (00747) 07:      ADDL(BW2,W$)             ;INCR M1
 A23 05EBC 4650000A (00748)          LOAD(BW1,C0])            ;YBA, A(1)
 A24 05EBE 40700000 (00749)          LOAD(BW3,M$$)            ;(UNUSED)
 A25 05EC0 4A700000 (00750)          LOAD(BW3,M$$)            ;(UNUSED)
 A26 05EC2 4C91002C (00751)          ADDB(BW1,BW2,TF)         ;OQ: A(M)
 A27 05EC4 4E910032 (00752)          SUBL(BW1,W$,TF)          ;OQ: A(M-1)
 A28 05EC6 50312854 (00753)          MOVB(BW3,BW2),JUMP(VK9)         ;M'=W$*(M-1) AND ENTER LOOP
                    (00754) ;INNER LOOP: L=1,2,...,(M-1)/2
 A29 05EC8 5231003A (00755) 08:      ADDL(BW3,W$)             ;ADD EXTRA W$ SUBTRACTED AT LOOP TEST
 A2A 05ECA 5491002E (00756)          ADDB(BW1,BW3,TF)         ;OQ: A(M-L)
 A2B 05ECC 56310032 (00757) VK9:     SUBL(BW3,W$)
 A2C 05ECE 58910026 (00758)          SUBB(BW1,BW3,TF)         ;OQ: A(L)
 A2D 05ED0 5A312904 (00759)          SUBL(BW3,2*W$),JUMPP(08)  ;JUMP IF NOT DONE ITERATION
 A2E 05ED2 5C012201 (00760)          SUBL(BW0,1),JUMPP(07)    ;JUMP IF MORE ITERATIONS TO DO
 A2F 05ED4 5E200030 (00761)          CLEAR(RO)
 A30 05ED6 60000020 (00762)          NOP(0)
                    (00763) ;
     05ED8          (00764)          END
           00005EBC (00765) VKAS$A=0C                         ;ASSIGN VALUE TO CHAIN ANCHOR
     05ED8 00000000 (00766) VKAS$I: DATA    5F'0.0'           ;BLOCK FOR CONSTRUCTED INSTRUCTIONS
    ...
           0000006C (00767) VKAS$Z=0L-VKASI                   ;COMPUTE MODULE SIZE ON BUS 1
```

```
* APU3-PITCH(Y,A,U,B,C,D)        COMPUTE AND CODE PITCH
*                                ALSO COMPUTE ONE-TAP COEFF
*                                AND OUTPUT 3 AUTOCORRELATION
*                                COEFF CENTERED AT THE PITCH LAG
* WHR 8/80
*
*
*  INPUT:
*      U      AUTOCORRELATION COEFFICIENTS(U(14),...,U(133))
*
*    !!!! NOTE: BUFFER U MUST BE COMPACT.!!!!
*
*      SC     CONSTANT = 14.0
*  OUTPUTS:
*      SA     PITCH LAG (IN SAMPLES)
*      SB     CODED PITCH(INTEGER)
*      SD     ONE-TAP COEFFICIENT
*      Y      3 AUTOCORRELATION COEFFICIENTS
*             CENTERED AT THE PITCH LAG
*             -U(PITCH LAG-1)
*             -U(PITCH LAG)
*             -U(PITCH LAG+1)
*
*  PROCEDURE:
*               DO SMAX(U) TO GET U(MAX) AND LOC OF MAX(=PITCH-14)
*               COMPUTE AND OUTPUT CODED PITCH=INTEGER OF MAX=>SB
*               COMPUTE AND OUTPUT PITCH LAG=MAX+14=>SA
*               COMPUTE INTEGER PITCH LAG FOR ADDRESS MOD
*               MODIFY APS ADDRESS
*               COMPUTE AND OUTPUT ONE-TAP COEFF =>SD
*               OUTPUT 3 AUTOCORRELATION COEFFS
*               CENTERED AT PITCH LAG ; U=>Y
*
*
*  INPUT STREAM: U(14),...,U(133),[FWI],(2**-15),SC(CONST=14.0),[FWI],
*             ,'MODIFIED INSTR',[FWI],U(0),U(PITCH LAG-1),U(PITCH LAG)
*             ,U(PITCH LAG+1)
*
*  OUTPUT STREAM: SB(CODED PITCH),SA(PITCH LAG=MAX+14),'INTEGER
*             PITCH','MODIFIED INSTR',SD(ONE TAP COEFF)
*             ,3 AUTOCORRELATION COEFF'S (Y(0),Y(1),Y(2)).
*
*  WHERE:
*             'INTEGER PITCH' IS USED BY APS OUTPUT PGM TO MODIFY
*                     APS INPUT PGM
*             'MODIFIED INSTR' IS APS INSTR TO BE MODIFIED.
*                     IT IS READ FROM BUS1, AND WRITTEN
*                     TO APS PSEUDO MEMORY.
*
*  REGISTER USAGE:
*    SMAX SECTION:
*      ALL A REGISTERS ARE USED
*      LOC OF MAX STORED IN A3
```

```
                        *       NONE OF THE M REGISTERS ARE USED EXCEPT M7
                        *       VALUE OF COEFF AT MAX LOC STORED IN M7
                        *    OTHERWISE:
                        *
                        *       A0 U(0)(SDIV) \                        M0 SDIV \
                        *       A1 SDIV,U(PITCH-1) \ U(PITCH)          M1 2**-15\
                        *       A2 U(MAX)/U(0) \                       M2 ---\---
                        *       A3 LOC U(MAX) \                        M3 --- \ ---
                        *       A4 CONST=14 \                          M4 PITCH LAG=MAX+14,SDIV \
                        *       A5 LAG*2**-15 \                        M5 LOC U(MAX),-1 \
                        *       A6 SDIV,U(PITCH+1) \                   M6 U(0)(SDIV) \
                        *       A7 MAX*2**-15 \                        M7 U(MAX) \
                        *
                        *  PITU - APU INITIALIZATION
                        *
                                EVEN                                START ON WORD BOUNDARY
    05EF2 0000                  DATA  PITU$SA                        START ADDRESS
    05EF3 005C                  DATA  PITU$SZ                        SIZE
                        *
                        PITU: BEGIN APU(PITU)                        START OF APU MODULE
                        *
                        *  DO SMAX ON U ARRAY (TERMINATE INPUT ON EFWI3)
                        *
A00 05EF4 00001600      PITU$SA: NOP \ K(1)                          ---            \ R=1
A01 05EF6 00F70094              MOV(IQA,A7)\ MOV(R,A4)         A7=U(14)       \ A4=1
A02 05EE0 00160096              MOV(ZERO,A6)\ MOV(R,A6)        A6=0           \ A6=1
A03 05EEA 16A016A0              K(2)                           R=2            \ R=2
A04 05EEC 000000F7              NOP \ MOV(IQA,A7)              ---            \ A7=U(15)
A05 05EEE 009502F5              MOV(R,A5) \ MOV(A5),R(A7)         A5=2        \ A5=2, R=U(15)
A06 05EF0 91160010              JUMPS(PITU$1,FWI)
                        *
A07 05EF2 00F00000      PITU$0:  MOV(IQA,A0) \ NOP                 A0=UE      \ ---
A08 05EF4 60F40097              MOV(A4),MAX(A7,A0) \ MOV(R,A7)  A4=LOC, R=MAX \ A7=MAX
A09 05EF6 011F0010              CALLS(PITU$3,T2)                 ---          \ IF NEW MAX
A0A 05EF0 000044A0              NOP \ ADD(A5,A4)               ---            \ R=LOC
A0B 05EFA 9116001B              JUMPS(PITU$4,FWI)                 EXIT IF UBS IS ODD
A0C 05EFC 009700F0              MOV(R,A7) \ MOV(IQA,A0)        A7=MAX         \ A0=UO
A0D 05EFE 011F0015              CALLS(PITU$2,T1)                IF NEW MAX    \ ---
A0E 05F00 44A060F4              ADD(A5,A4) \ MOV(A4),MAX(A7,A0)  R=LOC     \ A4=LOC,R=MAX
A0F 05F02 90160007              JUMPC(PITU$0,FWI)                   EXIT IF UBS IS EVEN
                        *
                        *                                      EXIT IF LOOP BOTTOM
A10 05F04 02E00000      PITU$1:  R(A7) \ NOP                      R=MAX       \ ---
A11 05F06 00910090              MOV(R,A1) \ MOV(R,EXO)         A1=MAX         \ EXO=MAX
A12 05F00 011F0010              CALLS(PITU$3,T2)                 ---          \ IF NEW MAX
A13 05F0A 00520000              MOV(EXI,A2) \ NOP              A2=MAX         \ ---
A14 05F0C 1000001F              JUMP(PITU$5)                     TO COMMON CLEAN UP
                        *
                        *                                     SUBROUTINE FOR LW SIDE
A15 05F0E 02000000      PITU$2: R(A4) \ NOP                      R=LOC        \ ---
A16 05F10 00960000              MOV(R,A6) \ NOP                 A6=LOC        \ ---
A17 05F12 A000A000              RETURN
                        *
```

```
PAGE   22:  [ABNO]<DCA16>ABU16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
             APU3-PITCH(Y,A,U,B,C,D)        COMPUTE AND CODE PITCH

                              *                                          SUBROUTINE FOR RH SIDE
  A18 05F14 00000200  PITUS3: NOP \ R(A4)                                 ---- \ R=LOC
  A19 05F16 00000096            NOP \ MOV(R,A6)                           --- \ A6=LOC
  A1A 05F18 1000A000            RETURN
                              *
                              *                                          IF EXIT LOOP MIDDLE
  A1B 05F1A 000002E0  PITUS4:  NOP \ R(A7)                                ---          \ R=MAX
  A1C 05F1C 00910098          MOV(R,A1) \ MOV(R,EXO)                    A1=MAX       \ EXO=MAX
  A1D 05F1E D11F0015          CALLS(PITUS2,T1)                            IF NEW MAX   \ ---
  A1E 05F20 00520000          MOV(EXI,A2)  \ NOP                        A2=MAX       \ ---
                              *
                              *                                          COMMON CLEAN UP
  A1F 05F22 622002C0  PITUS5:  MAX(A1,A2) \ R(A6)                          R=U-MAX        \ R=LOC(U-MAX)
  A20 05F24 000F0000          MOV(R,M7) \ NOP                           M7 = U(MAX)  \ ---
  A21 05F26 911E0025          JUMPS(PITUS6,T1)                            IF FROM RIGHT BOARD
  A22 05F28 02C00000          R(A6) \ NOP                               R=LOC(U-MAX) \ ---
  A23 05F2A 00930000          MOV(R,A3) \ NOP                           A3 = MAX \ ---
  A24 05F2C 10000027            JUMP(PITUS7)
  A25 05F2E 00000098  PITUS6:  NOP \ MOV(R,EXO)                            ---            \ EXO = MAX
  A26 05F30 00530000            MOV(EXI,A3) \ NOP                       A3 = MAX \ ---
                              *
                              *  END OF SMAX - MAX LOC IN A3 , MAX COEFF IN M7
                              *
                              *  COMPUTE CODED PITCH(MULT MAX LOC BY (2**-15) , THEN ALIGN)
                              *       (RESTART APS INPUT PGM BY CLEARING "WI")
                              *
  A27 05F32 20372037  PITUS7:  CLEAR(WI)                              ;RESTART APS INPUT PGM
  A28 05F34 00E90000            MOV(IQA,M1)\NOP                       ;M1=(2**-15)
  A29 05F36 02600000            R(A3)\NOP                             ;R=A3=LOC OF MAX
  A2A 05F38 000D0000            MOV(R,M5)\NOP                         ;M5=LOC OF MAX
  A2B 05F3A 84A00000            MUL(M1,M5)\NOP                        ;P=MAX*(2**-15)
  A2C 05F3C 00B70000            MOV(P,A7)\NOP                         ;A7=MAX*(2**-15)
  A2D 05F3E 3AE00000            ALIGN(A7)\NOP                         ;R=FXD CODED PITCH
                              *                                       ;(RIGHT JUST IN LHW)
  A2E 05F40 009C0000            MOV(R,DQ)\NOP                         ;DQ=S0=CODED PITCH
                              *
                              *  ADD 14 TO MAX LOC TO GET PITCH LAG (FLT POINT VALUE)
                              *
  A2F 05F42 00F40000            MOV(IQA,A4)\NOP                       ;INPUT CONSTANT = 14
  A30 05F44 44600000            ADD(A3,A4)\NOP                        ;R=MAX+14=PITCH LAG
  A31 05F46 009C0000            MOV(R,DQ)\ NOP                        ;DQ = SA = 14+MAX = PITCH LAG
                              *
                              *  GENERATE INTEGER PITCH LAG FOR ADDRESS MODIFICATION
                              *  (MULT BY (2**-15), THEN ALIGN)
                              *
  A32 05F48 000C0000            MOV(R,M4) \ NOP                       ; M4 <= PITCH LAG
  A33 05F4A 84000000            MUL(M1,M4) \ NOP                      ; P <= PITCH LAG*(2**-15)
  A34 05F4C 00050000            MOV(P,A5) \ NOP                       ; A5 <= PITCH LAG*(2**-15)
  A35 05F4E 3AA00000            ALIGN(A5) \ NOP                       ; R <= FXD PITCH LAG
                              *                                       ;(RGHT JUST IN L HW)
                              *
                              *  SEND TO BUS1 TO MODIFY BUS1 COPY OF APS INSTR.
                              *
```

```
A36 05F50 009C0000          MOV(R,OQ) \ NOP          ; OQ <= INTEGER PITCH
                    *
                    *  WAIT TIL BUS1 COPY IS MODIFIED, THEN READ
                    *  IT AND WRITE TO APS PSEUDO MEMORY.
                    *  (ALSO CLEAR WI SO APS INPUT PGM WILL CONTINUE.)
                    *  (IT WAS WAITING FOR WRITE TO COMPLETE.)
                    *
A37 05F52 90100037  PITU$W1: JUMPC(PITU$W1,OQE)      ;LOOP WHILE OUTPUT QUEUE NOT EMPTY
A38 05F54 00000000          NOP                      ;DELAY 1 FOR MEMORY WRITE
A39 05F56 20372037          CLEAR(WI)                ;NOW LET APS PROCEED
                    *
A3A 05F58 00FC0000          MOV(IQA,OQ) \ NOP        ; THEN READ MODIFIED APS INSTR
                    *                                ; AND WRITE TO APS PSEUDO MEM.
                    *
                    *  WAIT TIL APS COPY IS WRITTEN , THEN PROCEED APS INPUT PGM AGAIN
                    *
A3B 05F5A 9010003B  PITU$W2: JUMPC(PITU$W2,OQE)      ;LOOP WHILE OUTPUT QUEUE NOT EMPTY
A3C 05F5C 00000000          NOP                      ;DELAY 1 AS ABOVE
                    *
A3D 05F5E 20372037          CLEAR(WI)
                    *
                    *  SET UP REGS FOR DIVIDE: U(MAX)/U(0)
                    *  M7 ALREADY = U(MAX)
                    *  GET U(0) FROM IQ
                    *
A3E 05F60 00CF0000          MOV(IQ,M6) \ NOP          M6 = U(0) \ ---
A3F 05F62 00F80000          MOV(IQA,A0) \ NOP         A0 = U(0) \ ---
                    *
                    *  DO SDIV:  DIVIDE B/C
                    *  EXPECTS (IN LEFT ADM) B IN M7
                    *                        C IN M6
                    *                        C IN A0
                    *
A40 05F64 16A00000  PITU$0:  K(2) \ NOP                R=2
A41 05F66 00960000        MOV(R,A6) \ NOP          A6=2
A42 05F68 2E000000        RCP(A0) \ NOP            R1=1/C+DEL(=F0)(= 1/C + &)
A43 05F6A 00000000        MOV(R,M0) \ NOP          M0=F0
A44 05F6C 04400000        MUL(M0,M6) \ NOP         P1=F0*C
A45 05F6E 00810000        MOV(P,A1) \ NOP          A1=F0*C
A46 05F70 49C00000        SUB(A6,A1) \ NOP         R2=2-F0*C (= 1 - C&)
A47 05F72 008C0000        MOV(R,M4) \ NOP          M4=R2
A48 05F74 04000000        MUL(M0,M4) \ NOP         P2=F0*R2(=F1)(= 1/C - (&**2)C )
A49 05F76 00A00000        MOV(P,M0) \ NOP          M0=F1
A4A 05F78 04400000        MUL(M0,M6) \ NOP         P3=F1*C
A4B 05F7A 00810000        MOV(P,A1) \ NOP          A1=P3
A4C 05F7C 49C00000        SUB(A6,A1) \ NOP         R3=2-F1*C
A4D 05F7E 008C0000        MOV(R,M4) \ NOP          M4=R3
A4E 05F80 04000000          MUL(M0,M4) \ NOP       P4=F1*(2-F1*C)(=F2)(= 1/C - (&**4)(C**3)) (=1/C)
A4F 05F82 00A00000        MOV(P,M0) \ NOP          M0=F2
A50 05F84 04600000          MUL(M0,M7) \ NOP       P5=F2*B (=B/C)
A51 05F86 00820000          MOV(P,A2) \ NOP        A2 = TAP  \ ---
                    *
                    *  SDIV COMPLETE - NEGATE ONE-TAP COEFF AND OUTPUT
```

```
                         (00980) *
 A52 05F88 0A400000 (00981)        NEG(A2)\NOP               ;R=-U(MAX)/U(0)
 A53 05F8A 009C0000 (00982)        MOV(R,DQ)\NOP             ;OUTPUT NEGATED ONE-TAP COEFF
                         (00983) *
                         (00984) *
                         (00985) *  INPUT (AND OUTPUT) COEFFS CENTERED AT PITCH LAG
                         (00986) *
 A54 05F8C 00F10000 (00987)        MOV(IQA,A1)\NOP           ;A1=U(PITCH-1)\
 A55 05F8E 0A2000F1 (00988)        NEG(A1)\MOV(IQA,A1)       ;R=-U(PITCH-1)\A1=U(PITCH)
 A56 05F90 00F60A20 (00989)        MOV(IQA,A6)\NEG(A1)       ;A6=U(PITCH+1)\R=-U(PITCH)
 A57 05F92 009C0000 (00990)        MOV(R,DQ)\NOP             ;Y(0)=-U(PITCH-1)\
 A58 05F94 0AC0009C (00991)        NEG(A6)\MOV(R,DQ)         ;R=-U(PITCH+1)\Y(1)=-U(PITCH)
 A59 05F96 009C0000 (00992)        MOV(R,DQ)\NOP             ;Y(2)=-U(PITCH+1)
                         (00993) *
                         (00994) *
 A5A 05F98 20322032 (00995)        CLEAR(RA)
 A5B 05F9A 10000000 (00996)        JUMP(0)
                         (00997) *
           0000005C (00998)        PITU$SZ=$A-PITU$SA
     05F9C          (00999)        END
```

```
                         * APS3-PITCH(Y,A,U,B,C,D)
                         *
                         *
                         *  INPUT STREAM: U(14),...,U(133),[FWI],(2**-15),SC(CONST=14.0),
                         *                ,'MODIFIED INSTR',U(0),U(PITCH LAG-1),U(PITCH LAG)
                         *                ,U(PITCH LAG+1)
                         *
                         *  OUTPUT STREAM: SB(CODED PITCH),SA(PITCH LAG=MAX+14),'INTEGER
                         *                PITCH','MODIFIED INSTR',SD(ONE TAP COEFF)
                         *                ,3 AUTOCORRELATION COEFF'S (Y(PITCH LAG-1),Y(PITCH LAG)
                         *                ,Y(PITCH LAG+1)).
                         *
                         *  WHERE:
                         *                'INTEGER PITCH' IS USED BY APS OUTPUT PGM TO MODIFY
                         *                        APS INPUT PGM
                         *                'MODIFIED INSTR' IS APS INSTR TO BE MODIFIED.
                         *                        IT IS READ FROM BUS1, AND WRITTEN
                         *                        TO APS PSEUDO MEMORY.
                         *
                         *  HEADER BLOCK
                         *
                                EVEN
    05F9C 00006000              ADDR    PITS$I                  ; CONSTR INSTR BLK
    05F9E 00005FA6              ADDR    PITS$ + 2*PITS$S        ;SCALAR BLK
    05FA0 0004                  DATA    4                       ; NUMBER OF SCALARS
    05FA1 0070                  DATA    PITS$Z                  ; MODULE SIZE
    05FA2 00006002              ADDR    PITS$A                  ; PTR TO CHAIN ANCHOR
                                EVEN
                         *
                         *  END OF HEADER BLOCK
                         *
                         *  INPUT PROGRAM
                         *  REGISTER USAGE:
                         *      BR0     SA,U(BA)
                         *      BR1     SB,PITCH VALUE
                         *      BR2     SC,'MOD INSTR'
                         *      BR3     SD,COUNTER 14-2 &133-14-2,2**-15
                         *
                         *  START OF APS PROGRAM
                         *
                         PITS$:  BEGIN   APS(PITS)              ; START OF APS MODULE
                         *
                         *
 A00 05FA4 00202640             JSH(PITS$D,P2)                  ; SET OUTPUT PC
                         *
                         *  LOAD SCALARS
 A01 05FA6 02420000      PITS$S: LOAD(BR0,M$$(1))               ;SA ADDR
 A02 05FA8 04520000             LOAD(BR1,M$$(1))                ;SB ADDR
 A03 05FAA 06620000             LOAD(BR2,M$$(1))                ;SC ADDR
 A04 05FAC 00720000             LOAD(BR3,M$$(1))                ;SD ADDR
                         *
                         *  MOVE SCALAR ADDR TO OUTPUT REGS
                         *
```

```
A05 05FAE 0A010011 (01053)        MOVB(BW0,BR0)                ;BW0=SA
A06 05FB0 0C110013 (01054)        MOVB(BW1,BR1)                ;BW1=SB
A07 05FB2 0E310017 (01055)        MOVB(BW3,BR3)                ;BW3=SD
                   (01056) *
                   (01057) *  SCALARS SET , NOW START OUTPUT PROGRAM
                   (01058) *
A08 05FB4 10300030 (01059)        SET(RO)                      ;RUN OUTPUT PGN
                   (01060) *
                   (01061) *  GEN U(14)-U(133) ADDRESSES, THEN [PWI]
                   (01062) *
                   (01063) *   !!!! NOTE : BUFFER U MUST BE COMPACT. !!!!
                   (01064) *
                   (01065) *
A09 05FB6 12401000 (01066)        LOAD(BR0,[1])               ; LOAD U BASE ADDR, U(0)
A0A 05FB8 14500000 (01067)        LOAD(BR1,N$$)               ; LOAD UBS-1 (UNUSED)
A0B 05FBA 160A0000 (01068)        ADD(BR0,N$$)                ; ADD SPACING , U(1)
                   (01069) *
A0C 05FBC 18700000 (01070)        LOAD(BR3,14-3)              ; LOAD 14-3
A0D 05FBE 1A0A9000 (01071) B1:    ADD(BR0,[9])                ; CYCLE TILL 14TH
A0E 05FC0 1C390001 (01072)        SUBL(BR3,1),JUMPP(B1)       ; VALUE IS READY
                   (01073) *
A0F 05FC2 1E700076 (01074)        LOAD(BR3,133-14-1)          ;LOAD (133-14-1)
A10 05FC4 200A9006 (01075) B2:    ADD(BR0,[9],TF)             ;GEN U-ELEM ADDR
A11 05FC6 22391001 (01076)        SUBL(BR3,1),JUMPP(B2)        ;DO 14 - 132 ELEMENTS
A12 05FC8 2409137A (01077)        ADDL(BR0,W$,TF),JUMP($A+1)   ;AVOID APU/APS RACE
A13 05FCA 26300037 (01078)        SET(WI)                     ; SET WI.
A14 05FCC 28000020 (01079)        NOP(0)                       ;WAIT TIL APU PGN CLEARS IT
                   (01080) *
                   (01081) *  GEN (2**-15)
                   (01082) *
A15 05FCE 2AF203CE (01083)        LOAD(BR3,SVTSUN1(1),TF)     ; (2**-15)
                   (01084) *
                   (01085) *  GENERATE SC ADDR - CONST=14
                   (01086) *
A16 05FD0 2CA90015 (01087)        MOVB(BR2,BR2,TF)             ;GEN SC ADDR
                   (01088) *
                   (01089) *
                   (01090) *
                   (01091) *  WAIT FOR APU PGN TO SIGNAL THAT BUS1
                   (01092) *  COPY OF MODIFIED INSTR HAS BEEN
                   (01093) *  WRITTEN.
                   (01094) *
A17 05FD2 2E300037 (01095)        SET(WI)
A18 05FD4 30000020 (01096)        NOP(0)
                   (01097) *
                   (01098) *  GEN 'MODIFIED INSTR' ADDRESS.
                   (01099) *
                   (01100) *
A19 05FD6 32E75FE2 (01101)        LOAD(BR2,PITS$1(1),TF)      ; MODIFIED INSTR (BUS1 ADDR)
                   (01102) *
                   (01103) *  WAIT FOR APU PROGRAM TO SIGNAL THAT
                   (01104) *  MOIFIED INSTR IS COMPLETE
                   (01105) *
```

```
A1A 05FD8 34300037 (01106)        SET(WI)
A1B 05FDA 36000020 (01107)        NOP(0)
                   (01108) *  SET UP FOR OUTPUT OF AUTOCORRELATION COEFF
                   (01109) *    FIRST GEN U(0) FOR ONE -TAP COEFF
                   (01110) *
A1C 05FDC 3AF01010 (01111)        LOAD(BR3,[1],TF)              ; LOAD AND GEN U(0) ADDRESS
A1D 05FDE 3A600000 (01112)        LOAD(BR2,M$$)                 ;LOAD UBS-1 (UNUSED)
A1E 05FE0 3C320000 (01113)        SUB(BR3,M$$)                  ;SUB SPACING
                   (01114) *
                   (01115) *  !!!!!!!!!!!NEXT INSTR MODIFIED BY OUTPUT PGM!!!!!!!!!!!
          00005FE2 (01116) PITS$1 = $L
A1F 05FE2 3E500000 (01117)        LOAD(BR1,M$$)                 ; LOAD PITCH VALUE
A20 05FE4 40190020 (01118)        ADDB(BR1,BR1)                 ; DOUBLE PITCH FOR FW ADDRESSING
                   (01119) *
                   (01120) *  GEN ADDR FOR U(PITCH-1),U(PITCH),U(PITCH+1)
                   (01121) *
                   (01122) *    !!!!! NOTE : BUFFER U MUST BE COMPACT. !!!!!
                   (01123) *
                   (01124) *
A21 05FE6 42890020 (01125)        ADDB(BR3,BR1,TF)              ;GEN U(PITCH-1)
A22 05FE8 4489003A (01126)        ADDL(BR3,W$,TF)               ;GEN U(PITCH)
A23 05FEA 4689003A (01127)        ADDL(BR3,W$,TF)               ;GEN U(PITCH+1)
                   (01128) *
A24 05FEC 48200031 (01129)        CLEAR(RI)                     ; HALT INPUT
A25 05FEE 4A000020 (01130)        NOP(0)
                   (01131) *
                   (01132) *
                   (01133)
                   (01134) *
                   (01135) *  OUTPUT PROGRAM
                   (01136) *  REGISTER USAGE:
                   (01137) *        BW0     SA,Y(BA)
                   (01138) *        BW1     SB,'MOD INSTR','APS MEM'
                   (01139) *        BW2
                   (01140) *        BW3     SD
                   (01141) *
                   (01142) *
A26 05FF0 4C300032 (01143) PITS$0: SET(RA)                      ; TURN ON APU
                   (01144) *
                   (01145) *  FIRST, GENERATE SCALAR B AND A ADDRESSES
                   (01146) *
A27 05FF2 4F110012 (01147)        MOVB(BW1,BW1,TE)              ;GEN SB ADDR
A28 05FF4 50010010 (01148)        MOVB(BW0,BW0,TF)              ;GEN SA ADDR
                   (01149) *
                   (01150) *  NOW, GEN BUS1 ADDRESS OF INPUT PGM INSTR TO BE MODIFIED
                   (01151) *  ('INTEGER PITCH' WRITTEN INTO RIGHT HW OF INSTR.)
                   (01152) *
A29 05FF6 53525FE3 (01153)        LOAD(BW1,PITS$1+H$(1),TE)     ; SEND BS1 ADDR OF RHW OF INSTR
                   (01154) *
                   (01155) *  NOW WRITE MODIFIED INSTRUCTION TO APS MEMORY
                   (01156) *  VIA BUS1 PSEUDO MEMORY.
                   (01157) *
A2A 05FF8 54D3FFC0 (01158)        LOAD(BW1,APS$LA(1),TF)
```

```
                      (01159) *
                      (01160) *  GEN D SCALAR ADDR
                      (01161) *
  A2B 05FFA 56B10016  (01162)        MOVB(BW3,BW3,TF)           ; GEN SD ADDR
                      (01163) *
                      (01164) *  GENERATE AUTOCORRELATION COEFF ADDR
                      (01165) *
  A2C 05FFC 58C00020  (01166)        LOAD(BW0,[0],TF)           ; LOAD Y BASE ADDR, GEN Y(0) ADDR
  A2D 05FFE 5A600000  (01167)        LOAD(BW2,M$$)              ; LOAD YBS-1 (UNUSED)
  A2E 06000 5C0A0000  (01168)        ADD(BW0,M$$,TF)            ; GEN Y(1) ADDR
  A2F 06002 5E0A0006  (01169)        ADD(BW0,[0],TF)            ;GEN Y(2) ADDR
                      (01170) *
  A30 06004 60200030  (01171)        CLEAR(RO)                  ; HALT OUTPUT
  A31 06006 62000020  (01172)        NOP(0)
                      (01173) *
        00006002      (01174) PITS$A=0C                         ; CHAIN ANCHOR
                      (01175) *
      06000           (01176)        END
                      (01177) *
      06000 00000000  (01178) PITS$I  DATA    6F"0.0"
      ...
            00000070  (01179) PITS$Z=0L-PITS$                   ; MODULE SIZE
```

```
* NWLQ6(V,A,U,V,W)   WEINER-LEVINSON MATRIX SOL'N & P=6 CODING/QUANT.
*
*      NWLQ6-APU PROGRAM
;
;  THE CHANGES FROM NWLQ IN BBN300.NSO ARE:
;  1. CODE/QUANTIZE 6 COEFFICIENTS INSTEAD OF 8.
;  2. THE QUANTIZED K-VALUES ARE REGULAR FLT PT, NOT 16-BIT FIXED.
;  3. THEREFORE THE THRESH/VALUE TABLES ARE FLOATING.  SEPARATE,
;     NOT INTERLEAVED, THRESH/VALUE TABLES ARE USED.  THE VALUE
;     TABLE IS THE K-DECODING TABLE.
;
*
*
* WEINER LEVINSON DURBAN INVERSE (NWLF) FOLLOWED BY 6TH ORDER
*   REFLECTION COEFFICIENT CODING AND QUANTIZATION (KQUAN)
*
* MATHEMATICS
* SEE J. MAKHOUL,LINEAR PREDICTION REVIEW
* PROC. IEEE, VOL 63, PP566, APR 1975
*
* BUFFER DEFINITIONS FOR THE NWLF PART:
*
* V(K), AUTOCORRELATION COEFFICIENTS, V(K)=R(K)
*   V(K) MUST BE COMPACT 32 BIT FLOATING POINT
*   VSIZE NOT USED IN COMPUTATION, BUT
*   VSIZE > USIZE FOR VALID RESULTS.
*
* U(K),A(K) COEFFICIENTS
*   U(K),MUST BE COMPACT 32 BIT FLOATING POINT
*   U(K)=A(K-1),IE.A(0)=1 NOT IN BUFFER
*
* V(K), PARTIAL CORRELATIONS AND ERROR
*   V(K) MUST BE COMPACT 32 BIT FLOATING POINT
*   VSIZE=2*USIZE
*   V(K)=P(K-1), 0<=K<USIZE
*   V(K+USIZE)=E(K-1), USIZE<=K<VSIZE
*
* A,STABILITY TEST PARAMETER
* IF /P(N)/> CONTENTS OF (A), THEN
* FOR J>0
* A(N+J)=P(N+J)=0, E(N+J)=E(N)
*
* FOR THE KQUAN PART, OUTPUT BUFFERS:
*   THE W(K) BUFFER IS OVERWRITTEN BY THE QUANTIZED REFLECTION
*      (PARTIAL CORRELATION) COEFFICIENTS.
*
* W(K) = 6 CODED REFLECTION COEFFICIENTS.  W IS A COMPACT LONG (16 BIT)
*   FIXED POINT BUFFER.
*
*
* NWLF-APU INITIALIZATION
*
*
```

```
                         (01233)            EVEN                         START ON WORD BOUNDRY
     06014 0000          (01234)            DATA     MWLF$SA             START ADDRESS  ,
     06015 0065          (01235)            DATA     MWLF$SZ             SIZE
                         (01236) *
                         (01237) *
                         (01238) MWLF$APU  BEGIN APU(MWLFU)
           00000000      (01239) $A=0
                         (01240) *
A00 06016 00F70000 (01241) MWLF$SA MOV(IQA,A7)\NOP             A7=STABILITY TEST VALUE
A01 06018 000C000C (01242)         MOV(ZERO,M4)                M4=P(0)=0
A02 0601A 00CD0000 (01243)         MOV(IQ,M5)\NOP
A03 0601C 00F50000 (01244)         MOV(IQA,A5)\NOP             M5=A5=E(0)=R(0)
                   (01245) *
                   (01246) * MWLF-APU CALCULATION OF S(N)
                   (01247) * S(N)=R(N) + SUM[1,N-1] OF R(N-J)A(N-1,J)
                   (01248) * REGISTER CONTENTS AT START
                   (01249) * M4=P(N-1), M5=A5=E(N-1), A6=1, A7=TEST
                   (01250) *
A04 0601E 2EA02EA0 (01251) $1      RCP(A5)
A05 06020 00F20000 (01252)         MOV(IQA,A2)\NOP             A2=R(N)
A06 06022 00000000 (01253)         MOV(ZERO,M0)
A07 06024 04000400 (01254)         MUL(M0,M4)                   SETS P = 0
A08 06026 02490249 (01255)         MOV(M1), R(A2)              M1=RCP(E(N-1))
A09 06028 900A004F (01256)         JUMPC(MWLFNW,AF2)           FIRST TIME
                   (01257) *
A0A 0602A 904B0010 (01258)         JUMPC(MWLFSE,AF3),SET       EVEN OR ODD N
                   (01259) *
A0B 0602C 202B202B (01260)         CLEAR(AF3)
                   (01261) *
A0C 0602E 00EA0000 (01262) $2      MOV(IQA,M2)\NOP             M2=A(N-1,J)
A0D 06030 00EF0000 (01263)         MOV(IQA,M6)\NOP             M6=R(N-J)
A0E 06032 85508550 (01264)         MOV(A0), MUL(M2,M6)
A0F 06034 42124212 (01265)         MOV(A2), ADD(A0,A2)         S+A(N-1,J+1)R(N-J-1)
                   (01266) *
A10 06036 00EB0000 (01267) MWLFSE  MOV(IQA,M3)\NOP             M3=A(N-1,J+1)
A11 06038 00EF0000 (01268)         MOV(IQA,M7)\NOP             M7=R(N-J-1)
A12 0603A 85F185F1 (01269)         MOV(A1), MUL(M3,M7)
A13 0603C 42324232 (01270)         MOV(A2), ADD(A1,A2)         S+A(N-1,J)R(N-J)
                   (01271) *
A14 0603E 9016000C (01272)         JUMPC($2,FWI)
                   (01273) *
A15 06040 00000000 (01274)         MOV(P,A0)
A16 06042 42124212 (01275)         MOV(A2), ADD(A0,A2)
                   (01276) *
A17 06044 20372037 (01277) MWLFRE  CLEAR(WI)
                   (01278) *
                   (01279) *
                   (01280) * MWLF-APU CALCULATION OF P(N)
                   (01281) * P(N)=-S(N)/E(N-1)
                   (01282) * REGISTER CONTENTS AT START
                   (01283) * M4=P(N-1), A5=M5=E(N-1), A6=1, A7=TEST
                   (01284) * M1=RCP[E(N-1)], R=S(N)
                   (01285) *
```

```
 .  .   (00*0)<0CA16>00N16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
        MULAS6(V,A,U,V,W)   WEINER-LEVINSON MATRIX SOL'N & P=6 CODING/QUANT.

                     (01286) *
A18 06046 04A004A0 (01287)           MUL(M1,M5)                 E* RCP(E)
A19 06048 16AA16AA (01288)           MOV(M2), K(2)              M2=S(N)
A1A 0604A 00000000 (01289)           MOV(P,A0)                  A0=E* RCP(E)
A1B 0604C 40314031 (01290)           MOV(A1), SUB(A1,A0)        A1=2
A1C 0604E 160F160E (01291)           MUV(M6), K(1)              M6=2-E* RCP(E)
A1D 06050 04C004C0 (01292)           MUL(M1,M6)
A1E 06052 00A900A9 (01293)           MOV(P,M1)                  M1=A4=EIR,
A1F 06054 04B404B4 (01294)           MOV(A4), MUL(M1,M5)        EIR APPRO(1/E) TO 2^(-11)
A20 06056 00800080 (01295)           MOV(P,A0)
A21 06058 40D640D6 (01296)           MOV(A6), SUB(A6,A0)        A6=1
A22 0605A 000E000E (01297)           MOV(R,M6)                  M6=1-EIR* E
A23 0605C 04C004C0 (01298)           MUL(M1,M6)
A24 0605E 00800080 (01299)           MOV(P,A0)                  A0=EIR(1-E* EIR)
A25 06060 44004400 (01300)           ADD(A0,A4)
A26 06062 000E000E (01301)           MOV(R,M6)                  R=1/E(N-1), FULL PRECISION
A27 06064 A540A540 (01302)           MULN(M2,M6)
A28 06066 00B400B4 (01303)           MOV(P,A4)
A29 06068 02000200 (01304)           R(A4)      P(N)
A2A 0606A 00A800A8 (01305)           MOV(P,M0)                  M0=M4=A4=P(N)
A2B 0606C 00AC00AC (01306)           MOV(P,M4)
A2C 0606E 009C0000 (01307)           MOV(R,DQ)\NOP              DQ = P(N) (FIXED) = A(N,N)
                     (01308) *
                     (01309) *
                     (01310) *
                     (01311) * MWLF-APU CALCULATION OF A(N,J)
                     (01312) * A(N-1,J)=A(J)
                     (01313) * A(N,J)=A(J)+P(N)A(N-J)
                     (01314) * A(N,N-J)=A(N-J)+P(N)A(J)
                     (01315) * LOOP COMPUTES BOTH A(N,J) AND A(N,N-J)
                     (01316) * FOR N=2M, A(N,M) COMPUTED TWICE
                     (01317) *
                     (01318) * REGISTER CONTENTS AT START
                     (01319) * M0=M4=A4=P=P(N), M5=A5=E(N-1), A6=1, A7=TEST
                     (01320) *
                     (01321) * DATA SEQUENCE
                     (01322) * INPT, [A(N-J+1),A(N-J),A(J)] FOR [1,N/2],WI, A(N/2+1)
                     (01323) * OTPT, DUMP, DUMP [A(N,J), A(N,N-J)]
                     (01324) *
                     (01325) *
A2D 06070 904A0051 (01326)           JUMPC(MWLFAW,AF2), SET      FIRST TIME AF2 CLEAR
                     (01327) *
A2E 06072 00F30000 (01328)           MOV(IQA,A3)\NOP            A3=A(N-J+1)
                     (01329) *
A2F 06074 00EA0000 (01330) #3        MOV(IQA,M2)\NOP            M2=A(N-J)
A30 06076 05118511 (01331)           MOV(A1), MUL(M2,M4)
A31 06078 433C0000 (01332)           MOV(DQ), ADD(A1,A3)\NOP    DQ=A(J-1)+P(N)A(N-J+1)
                     (01333) *                                   A(N-J+1)+P(N)A(J-1)
A32 0607A 00D20000 (01334)           MOV(IQ,A2)\NOP             A2=M3=A(J)
A33 0607C 00EB0000 (01335)           MOV(IQA,M3)\NOP
A34 0607E 05900590 (01336)           MOV(A0), MUL(M3,M4)
A35 06080 421C0000 (01337)           MOV(DQ), ADD(A0,A2)\NOP    DQ=A(N-J+1)+P(N)A(J-1)
A36 06082 00F30000 (01338)           MOV(IQA,A3)\NOP
```

```
            MWLQ6(Y,A,U,V,W)   WEINER-LEVINSON MATRIX SOL'N & P=6 CODING/QUANT.

A37 06004 00000000 (01339)          NOP
A38 06006 9016002F (01340)          JUMPC(03,FWI)
                   (01341) *
A39 06008 20372037 (01342) MWLFAE   CLEAR(WI)
A3A 0600A 04110411 (01343)          MOV(A1), MUL(M0,M4)
A3B 0600C 433C0000 (01344)          MOV(DQ), ADD(A1,A3)\NOP
                   (01345) *
                   (01346) *
                   (01347) * MWLF-APU CALCULATION OF E(N)
                   (01348) * E(N)=[1-P(N)P(N)]E(N-1)
                   (01349) *
                   (01350) * REGISTER CONTENTS AT START
                   (01351) *
                   (01352) * M0=M4=A4=P(N), M5=A5=E(N-1), A6=1, A7=TEST
                   (01353) * P=P(N)P(N),R=P(N)=A(N,N)
                   (01354) * DATA SEQUENCE
                   (01355) * OTPT  P(N), E(N)
A3C 0600E 00000000 (01356)          MOV(P,A0)                   A0=P(N)P(N)
A3D 06090 40DC0000 (01357)          MOV(DQ), SUB(A6,A0)\NOP     DQ=A(N,N)
A3E 06092 020A020A (01358)          MOV(M2), R(A4)              M2=1-P(N)P(N)
A3F 06094 05200520 (01359)          MUL(M2,M5)                  E(N-1)* [1-P(N)P(N)]
A40 06096 74FC0000 (01360)          MOV(DQ), MAXABS(A7,A4)\NOP    DQ=P(N)
A41 06098 00000000 (01361)          MOV(R,NULL)                 T1 SET IF /P(N)/>TEST
A42 0609A 00BC0000 (01362)          MOV(P,DQ)\NOP               DQ=E(N)
A43 0609C 20892089 (01363)          SETTG(AF1)                  AF1 SET IF /P(N)/>TEST
A44 0609E 00B50005 (01364)          MOV(P,A5)
A45 060A0 00AD00AD (01365)          MOV(P,M5)
A46 060A2 20482048 (01366)          SET(AF0)                    LET APS/OTPT GO
A47 060A4 911D004D (01367)          JUMPS(MWLFD,FI)                GO BACK UNLESS DONE
                   (01368) *
A48 060A6 901E0004 (01369)          JUMPC(01,T1)
                   (01370) *
                   (01371) *
                   (01372) * MWLF-APU ABORT ROUTINE
                   (01373) * P REGISTER=E(N)
                   (01374) *
                   (01375) *
A49 060A8 001C0000 (01376) 04       MOV(ZERO,DQ)\NOP            A(N+J)=0
A4A 060AA 001C0000 (01377)          MOV(ZERO,DQ)\NOP            P(N+J)=0
A4B 060AC 00BC0000 (01378)          MOV(P,DQ)\NOP               E(N+J)=E(N)
A4C 060AE 901C0049 (01379)          JUMPC(04,EO)
                   (01380) *
A4D 060B0 20322032 (01381) MWLFD    CLEAR(RA)                ;APU HALTS HERE AT END OF MWLD,
A4E 060B2 10000053 (01382)          JUMP(KQUAN)              ; THEN GOES TO KQUAN WHEN RESTARTED
                   (01383) ;                                 ; BY THE APS PROGRAM
                   (01384) ;
A4F 060B4 9016004F (01385) MWLFRW   JUMPC(MWLFRW,FWI)
A50 060B6 100F0017 (01386)          JUMP(MWLFRE)
A51 060B8 00E00000 (01387) MWLFAW   MOV(IQA,NULL)\NOP
A52 060BA 10000039 (01388)          JUMP(MWLFAE)
                   (01389) ;END OF MWLD APU PROGRAM
                   (01390) ;
                   (01391) ;
```

```
                     ;
                     ;KQUAN: QUANTIZE AND CODE 6 REFLECTION COEFFICIENTS
                     ;  J. WOLF  7/8/80
                     ;
                     ;INPUT AND OUTPUT STREAMS:
                     ;IQ:     2**-15,
                     ;        FOR EACH COEFF: K(I), (CTHKI(N),DVLKI(N), N=1,...,LENGTH)[WI]
                     ;        APS SETS AF0 ON LAST K(I) (EO WILL BE TOO LATE).
                     ;OQ:     (FLT PT VALUE(I), FXD PT CODE(I), I=1,...,6)
                     ;
                     ;TABLE LENGTHS FOR K(I) ARE 64, 32, 16, 16, 16, 16
                     ;APU REGISTER USAGE:
                     ;  A0: K(I)              CODE CTR
                     ;  A1: CTHKI(N)          2**-15
                     ;  A2: DVLKI(N)
                     ;
A53 060BC 000000F1   KQUAN:  NOP \ MOV(IQA,A1)           ; \ 2**-15 FOR FXD PT COUNTING
                     ;LOOP FOR EACH NEW COEFFICIENT TO BE QUANTIZED/CODED
A54 060BE 00F00010   01:     MOV(IQA,A0) \ MOV(ZERO,A0)      ;INPUT COEFF \ INIT CODE=0
A55 060C0 00000200           NOP \ R(A0)
                     ;INNER LOOP
A56 060C2 00F10000   02:     MOV(IQA,A1) \ NOP               ;INPUT CTH(N)
A57 060C4 49004110           SUB(A0,A1) \ MOV(A0),ADD(A0,A1)  ;(K-CTH) \ UPDATE CODE, CODE+1
A58 060C6 00F20000           MOV(IQA,A2) \ NOP               ;INPUT DVL(N)
A59 060C8 00000000           MOV(R,NULL) \ NOP               ;ENSURE T1 SET
A5A 060CA 9116005F           JUMPS(KQ$DONE,FWI)              ;JUMP IF DONE TABLE
A5B 060CC 901F0056           JUMPC(02,T1)                    ;JUMP IF K>CTH (KEEP LOOKING)
                     ;FOUND THE RIGHT QUANTIZATION BIN.  NOW READ IN THE REST OF TABLE
A5C 060CE 0RE00000   03:     MOV(IQA,NULL) \ NOP
A5D 060D0 00000000           NOP                             ;DELAY TO ALLOW FWI TO SET
A5E 060D2 9016005C           JUMPC(03,FWI)
A5F 060D4 20372037   KQ$DONE: CLEAR(WI)                      ;LET APS INPUT PGM PROCEED
A60 060D6 02403A00           R(A2) \ ALIGN(A0)               ;QUANT VALUE \ FIX CODE VALUE
A61 060D8 089C009C           MOV(R,OQ) \ MOV(R,OQ)           ;OUTPUT QUANT, CODE VALUES
A62 060DA 90000054           JUMPC(01,AF0)                   ;AF0 IS SET IF THIS WAS FINAL COEFF
A63 060DC 20322032           CLEAR(RA)                       ;HALT APU
A64 060DE 10000000           JUMP(0)
                     ;
         00000065    MWLF$SZ=0A-MWLF$SA
                     ;
    060E0                    END
```

```
                      (01433) * MWLQ6-APS PROGRAM
                      (01434) *
                      (01435) *
                      (01436)            EVEN          START ON WORD BOUNDRY
    060E0 000061E2 (01437)            ADDR MWLF$I   POINTER TO CONSTRUCTED INSTRUCTION BLOCK
    060E2 000060EC (01438)            ADDR MWLF$APS+2*MWLF$S
    060E4 0001     (01439)            DATA 1
    060E5 010A     (01440)            DATA MWLF$Z   MODULE SIZE
    060E6 000061D0 (01441)            ADDR MWLF$A   SUBSTITUTABLE ARGUMENTS CHAIN ANCHOR
                      (01442)            EVEN          END ON WORD BOUNDRY
                      (01443) *
                      (01444) *
                      (01445) *
                      (01446) * WEINER - APS INITIALIZATION ROUTINES
                      (01447) *
                      (01448) MWLF$APS  BEGIN APS(MWLFS)
                      (01449) *
A00 060F8 00000260 (01450)          JUMP(MWLF$S)            ;RUN MWLF
A01 060FA 02004E60 (01451)          JUMP(KQ$I)      ;AFTER MWLF, SPECIAL SUPPORT NOP'S THE ABOVE
                      (01452) ;                          ;  INSTRUCTION, SO WE NOW RUN KQ$I
                      (01453) ;
A02 060EC 04E20000 (01454) MWLF$S  LOAD(BR2,M$$(1),L,TF)       INPUT STABILITY TEST
A03 060EE 06300032 (01455)          SET(RA)
                      (01456) *
A04 060F0 08C02000 (01457)          LOAD(BR0,[2],L,TF)          INPT R(0)
A05 060F2 0A500000 (01458)          LOAD(BR1,M$$)               NOT USED
A06 060F4 0C0A0000 (01459)          ADD(BR0,M$$)                BR0=R(1)
A07 060F6 0E500000 (01460)          LOAD(BR1,0)
A08 060F8 10010013 (01461)          MOVB(BW0,BR1)               BW0=0=2(N-1)
A09 060FA 1270100A (01462)          LOAD(BR3,[1],L)             BR3=A(1)
A0A 060FC 14500000 (01463)          LOAD(BR1,M$$)               BR1=ASIZE-1, NOT USED
A0B 060FE 16320000 (01464)          SUB(BR3,M$$)                BR3=A(0)
                      (01465) *
                      (01466) *
                      (01467) * MWLF - APS S(N) INPUT
                      (01468) * REGISTER CONTENTS AT START
                      (01469) * BR0=R(1), BW0=2(N-1), BR3=A(0)=AB-2
                      (01470) *
                      (01471) *
A0C 06100 10890020 (01472) 01      ADDB(BR0,BW0,TF)            INPT R(N)
A0D 06102 1A190010 (01473)          MOVB(BR1,BW0)               BR1=2(N-1)
A0E 06104 1C290017 (01474)          MOVB(BR2,BR3)
A0F 06106 1E1915F2 (01475)          SUBL(BR1,2),JUMPN(MWLSC)
A10 06108 20001260 (01476)          JUMP(MWLSE)
                      (01477) *
A11 0610A 22820002 (01478) 02      SUB(BR0,2,TF)               INPT R(N-J)
                      (01479) *
A12 0610C 24AA0002 (01480) MWLSE   ADD(BR2,2,TF)              INPT A(J)
A13 0610E 261911B2 (01481)          SUBL(BR1,2), JUMPP(02)
                      (01482) *
A14 06110 28891572 (01483)          SUBL(BR0,2,TF),JUMP(MWLSC)   INPT R(1)
                      (01484) *
A15 06112 2A3A0037 (01485) MWLSC   SET(WI)
```

```
A16 06114 2C000020              NOP(0)                      ;!!!ADDED BY KFIELD 11/79!!!
A17 06116 2F01003A              ADDL(BW0,2)                  BW0=2N
                       *
                       *
                       *
                       * MWLF = APS  A(N,J)  CALCULATION INPT
                       * REGISTER CONTENTS AT START
                       * BR0=R(1), BW0=2N
                       * BR2=A(N-1), BR3=A(0)=AB-2
                       *
                       *
A18 06118 302A0002              ADD(BR2,2)                   BR2=A(N)
A19 0611A 32210015              MOVB(BW2,BR2)
A1A 0611C 34202A40              JSN(MWLOT,P2)                SET UP OUTPUT
                       *
A1B 0611E 36191F50              MOVB(BR1,BW0), JUMP(MWLAE)   BR1=2N
                       *
A1C 06120 38AA0000      03      ADD(BR2,0,TF)                A(N-1,N-J+1)INPT
A1D 06122 3AA20002              SUB(BR2,2,TF)                A(N-1,N-J)INPT
A1E 06124 3CBA0002              ADD(BR3,2,TF)                A(N-1,J)INPT
                       *
A1F 06126 3E191CB4      MWLAE   SUBL(BR1,4), JUMPP(03)
                       *
A20 06128 4070102E      MWLT    LOAD(BR3,[1],L)              BR3=A(1)=AB
A21 0612A 42500000              LOAD(BR1,M$$)                ASIZE=N,DONE
A22 0612C 44320000              SUB(BR3,M$$)                 BR3=A(0)=AB-2
A23 0612E 46190020              ADDB(BR1,BR1)                BR1=2N-2, MWDONE
A24 06130 48A92570              ADDL(BR2,0,TF), JUMP(0A+1)        CLEANUP OF A(N) CALC*
A25 06132 4A300037              SET(WI)
A26 06134 4C000020              NOP(0)
A27 06136 4E190CA0              SUBB(BR1,BW0), JUMPP(01)
A28 06138 502026F1              JUMPS(0A-2,RI),CLEAR
A29 0613A 52000020              NOP(0)                      ;!!!ADDED BY KFIELD 11/79!!!
                       *
                       *
                       * MWLF - APS OUTPUT
                       *
                       * REGISTER CONTENTS AT START
                       * RW0=2N, BW2=A(N)
                       *
                       *
A2A 0613C 54300030      MWLOT   SET(RO)
A2B 0613E 56810014              MOVB(BW3,BW2,TF)             BW3=A(N), OUTPUT
A2C 06140 58210020              SUBB(BW2,BW0)                BW2=A(0)
A2D 06142 5A020794              LOAD(BW1,DMY$(1),TF)       OTPT DUMP
A2E 06144 5C910030              ADDL(BW1,0,TF)               OTPT DUMP
A2F 06146 5F113250              MOVB(BW1,BW0), JUMP(MWLOE)   BW1=2N
                       *
A30 06148 60AA0002      04      ADD(BW2,2,TF)                OTPT A(N,J)
A31 0614A 62B20002              SUB(BW3,2,TF)                OTPT A(N,N-J)
                       *
A32 0614C 641130B4      MWLOE   SUBL(BW1,4),JUMPP(04)
A33 0614E 66600026              LOAD(BW2,[0],L)              BW2=P(1)=PB
```

```
 A34 06150 60500000 (01539)          LOAD(BW1,M$$)
 A35 06152 6A220000 (01540)          SUB(BW2,M$$)              BW2=P(0)=PB-2
 A36 06154 6C1A0001 (01541)          ADD(BW1,1)                BW1=PSIZE=2ASIZE
 A37 06156 6FA10020 (01542)          ADDB(BW2,BW0,TF)          OTPT P(N)
 A38 06158 70A1002A (01543)          ADDB(BW2,BW1,TF)          OTPT K(N)
                    (01544) *
 A39 0615A 72000020 (01545) 05       NOP(0)
 A3A 0615C 740039A0 (01546)          JUMPC(05,AF0)             WAIT ON APU
                    (01547) *
 A3B 0615E 76203C60 (01548)          JUMP(0A+1,AF0),CLEAR
 A3C 06160 7800 4CA9 (01549)         JUMPC(MWLOD,AF1)          JUMP TO MWDONE IF NO ABORT
                    (01550) *
 A3D 06162 7A200031 (01551)          CLEAR(RI)
 A3E 06164 7C000020 (01552)          NOP(0)               ;!!!ADDED BY KFIELD 11/79!!!
 A3F 06166 7E390010 (01553)          MOVB(BR3,BW0)             BR3 = 2N
 A40 06168 80290010 (01554)          MOVB(BR2,BW0)             BR2 = 2N
 A41 0616A 82390022 (01555)          SUBB(BR3,BW1)             BR3=2N-ASIZE
 A42 0616C 84390031 (01556)          SUBL(BR3,1)               BR3=2N-ASIZE-1
                    (01557) *
 A43 0616E 86401020 (01558)          LOAD(BW0,[1])             BW0 = A(1)
 A44 06170 88700000 (01559)          LOAD(BW3,M$$)             NOT USED
 A45 06172 8A020000 (01560)          SUB(BW0,M$$)              BW0 = A(0)
 A46 06174 8C0148 6D (01561)         ADDB(BW0,BR2),JUMP(MWLOC)  BW0 = A(N)
                    (01562) *
 A47 06176 8E0A0002 (01563) 06       ADD(BW0,2,TF)             OTPT A(N+J)=0
 A48 06178 90210022 (01564)          SUBB(BW2,BW1)             BW2=P(N+J-1)
 A49 0617A 92AA0002 (01565)          ADD(BW2,2,TF)             OTPT P(N+J)=0
 A4A 0617C 94A1002A (01566)          ADDB(BW2,BW1,TF)          OTPT K(N+J)=K(N)
 A4B 0617E 963947FA (01567) MWLOC    ADDL(BR3,2), JUMPN(06)
                    (01568) *
 A4C 06180 98200030 (01569) MWLOD    CLEAR(RO)
 A4D 06182 9A000020 (01570)          NOP(0)
                    (01571) ;
                    (01572) ;
                    (01573) ;APS PROGRAM FOR QUANTIZATION AND CODING.  THIS IS STARTED
                    (01574) ;  BY THE MWLQ$SSM SPECIAL SUPPORT MODULE AFTER MWLD FINISHES.
                    (01575) ;J.J.WOLF, 7/8/80
                    (01576) ;
                    (01577) ;INPUT AND OUTPUT STREAMS:
                    (01578) ;IQ:     2**-15,
                    (01579) ;        FOR EACH COEFF: K(I), (CTNKI(N),DVLKI(N), N=1,...,LENGTH)[NI]
                    (01580) ;        APS SETS AF0 ON LAST K(I) (EO WILL BE TOO LATE).
                    (01581) ;OQ:     (FLT PT VALUE(I), FXD PT CODE(I), I=1,...,6)
                    (01582) ;
                    (01583) ;INPUT PROGRAM
                    (01584) ;  BR0: Y(I) = UNQUANTIZED K(I)
                    (01585) ;  BR1: THRESH(N)
                    (01586) ;  BR2: VALUE(N)
                    (01587) ;  BR3: TABLE LENGTH COUNTER
                    (01588) ;
 A4E 06184 9C207040 (01589) KQSI:    JSN(KQSO,P2)         ;SET AND START OUTPUT PROGRAM
 A4F 06186 9E300030 (01590)          SET(RO)
 A50 06188 A0106640 (01591)          JSN(KQ$SUB,P1)       ;SET UP P1 TO USE KQ$SUB AS A SUBROUTINE
```

```
A51 0610A A2C203CE (01592)          LOAD(BR0,SVTSUM1(1),TF)  ;GEN ADR FOR 2**-15
A52 0610C A440001E (01593)          LOAD(BR0,C0])            ;YBA: UNQUANTIZED K(I)
A53 0610E A6700000 (01594)          LOAD(BR3,M$$)            ;(UNUSED) VBS-1
A54 06190 A8020000 (01595)          SUB(BR0,M$$)             ;PREPARE TO INCREMENT LATER
A55 06192 AA52902E (01596)          LOAD(BR1,CTHK1-W$(1))    ;BR1 POINTS TO THRESHHOLD TABLE -1
A56 06194 AC679A2E (01597)          LOAD(BR2,DVLK1-W$(1))    ;BR2 POINTS TO VALUE(N) TABLE -1
A57 06196 AE70003E (01598)          LOAD(BR3,64-2)
A58 06198 B0040020 (01599)          NOP(C)                   ;"CALL" KQ$SUB FOR K1, TABLE LENGTH 64
A59 0619A B270001E (01600)          LOAD(BR3,32-2)
A5A 0619C B4040020 (01601)          NOP(C)                   ;K2,TABLE LENGTH 32
A5B 0619E B670000E (01602)          LOAD(BR3,16-2)
A5C 061A0 B0040020 (01603)          NOP(C)                   ;K3, TABLE LENGTH 16
A5D 061A2 BA70000E (01604)          LOAD(BR3,16-2)
A5E 061A4 BC040020 (01605)          NOP(C)                   ;K4, TABLE LENGTH 16
A5F 061A6 BE70000E (01606)          LOAD(BR3,16-2)
A60 061A8 C0040020 (01607)          NOP(C)                   ;K5, TABLE LENGTH 16
A61 061AA C270000E (01608)          LOAD(BR3,16-2)
A62 061AC C4300028 (01609)          SET(AF0)                 ;SIGNAL APU THIS IS THE LAST ONE
A63 061AE C6040020 (01610)          NOP(C)                   ;K6, TABLE LENGTH 16
A64 061B0 C0200031 (01611)          CLEAR(RI)                ;WE'RE DONE
A65 061B2 CA000020 (01612)          NOP(0)
                   (01613) ;
                   (01614) ;"SUBROUTINE" TO GENERATE INPUT ADRS FOR ONE COEFFICIENT, NAMELY
                   (01615) ;  THE UNQUANTIZED K(I), THEN THE THRESH(N), VALUE(N) TABLE VALUES.
                   (01616) ;UPON ENTRY, BR0=K(I-1), BR1=THRESH(-1), BR2=VALUE(-1), BR3=LNGTH-2
                   (01617) ;UPON EXIT, BR0=K(I), BR1=THRESH(LAST), BR2=VALUE(LAST)
                   (01618) ;
A66 061B4 CC000020 (01619) KQ$SUB: NOP(0)                  ;FOR JUMP AND JSR
A67 061B6 CE09003A (01620) 03:     ADDL(BR0,W$,TF)          ;GEN K(I)
A68 061B8 D099003A (01621) 01:     ADDL(BR1,W$,TF)          ;GEN THRESH(N)
A69 061BA D2A9003A (01622)         ADDL(BR2,W$,TF)          ;GEN VALUE(N)
A6A 061BC D4396001 (01623)         SUBL(BR3,1),JUMPP(01)    ;COUNT ENTRY AND JUMP IF MORE TABLE
A6B 061BE D699003A (01624)         ADDL(BR1,W$,TF)          ;DO LAST PAIR, THEN SET WI QUICKLY TO
A6C 061C0 D0A9607A (01625)         ADDL(BR2,W$,TF),JUMP(0A+1)  ;AVOID APS-APU RACE
A6D 061C2 DA300037 (01626)         SET(WI)                  ;SIGNAL END OF TABLE TO APU
A6E 061C4 DC040020 (01627)         NOP(C)                   ;RETURN FROM SUBR - JUMP(KQ$SUB,C)
A6F 061C6 DE006760 (01628)         JUMP(03)                 ; DOESN'T WORK!
                   (01629) ;
                   (01630) ;OUTPUT PROGRAM
                   (01631) ;  BW0: W(I) FLT PT QUANTIZED K(I)
                   (01632) ;  BW1: W(I) FIXED POINT CODED K(I)
                   (01633) ;  BW2: LOOP COUNTER, COUNTS 6 OUTPUT PAIRS
                   (01634) ;
A70 061C8 E0300032 (01635) KQSO:   SET(RA)
A71 061CA E240103E (01636)         LOAD(BW0,C1])            ;OUTPUT ARRAY (W)
A72 061CC E4600000 (01637)         LOAD(BW2,M$$)            ;(UNUSED) WBS-1
A73 061CE E6020000 (01638)         SUB(BW0,M$$)             ;PREPARE TO INCR LATER
A74 061D0 E0503006 (01639)         LOAD(BW1,C3])            ;OUTPUT ARRAY W
A75 061D2 EA600000 (01640)         LOAD(BW2,M$$)            ;(UNUSED) WBS-1
A76 061D4 EC120000 (01641)         SUB(BW1,M$$)             ;PREPARE TO INCR LATER
A77 061D6 EE600005 (01642)         LOAD(BW2,5)              ;NUMBER-1 OF DOUBLE OUTPUTS
A78 061D8 F001003A (01643) 02:     ADDL(BW0,W$,TF)          ;KVALUE(N)
A79 061DA F3110039 (01644)         ADDL(BW1,H$,TE)          ;KCODE(N)
```

```
A7A 061DC F42170B1 (01645)            SUBL(RW2,1),JUMPP($2)   ;DONE?
A7B 061DE F6200030 (01646)            CLEAR(R0)               ;YES
A7C 061E0 FA000020 (01647)            NOP(0)
                   (01648) *
                   (01649) *
          000061DE (01650) MWLFSA=$C
                   (01651) *
    061E2          (01652)            END
                   (01653) *
                   (01654) *              STORAGE FOR CONSTRUCTED INSTRUCTIONS
                   (01655) *
    061F2 00000000 (01656) MWLFSI     DATA 8F"0.0"
     ...
                   (01657) *
          0000010A (01658) MWLFSZ=$L-MWLFSAPS
```

```
                   (01659) * NWLQSSSM - SPECIAL INTERMEDIATE SUPPORT FOR NWLQ6
                   (01660) *
                   (01661) NWLQSSSM:
    061F2 066061F0 (01662)           CALL    R6,NWSSS1
    061F4 0F70     (01663)           RET
    061F5 0000FB1  (01664)           JMP APSDONE
                   (01665) *
                   (01666) *
    061F7 0000     (01667)           EVEN
    061F8 F0606202 (01668) NWSSS1:   MOVMR R6,INST
    061FA F0706203 (01669)           MOVMR R7,INST+1$
    061FC 8461FFC0 (01670)           MOVRML R6,APS$LA          ;LOAD NOP(0) INTO APS LOC 0
    061FE F063FFCE (01671)           MOVLM $31,SYS$FLGS        ;SET RI
    06200 0E70     (01672)           RETURN
                   (01673) ;
    06201 0000     (01674)           EVEN
                   (01675) INST:     BEGIN APS(FOO)
A00 06202 00000020 (01676)           NOP(0)                    ;THIS INSTR GETS PUT AT $A=0
    06204          (01677)           END
```

```
' NEWL(Y,I,U,N,V,W)
; CORRELATION MATRIX EQUATION, WEINER-LEVINSON SOLUTION
;
; SEPTEMBER 14,1979
;
;
; BUFFER USAGE
; Y(K), OUTPUT VECTOR
; U(K), INPUT VECTOR
; V(K)=R(K)=R(-K),CORRELATION MATRIX ELEMENTS
; W(K), WORKING BUFFER
; (I),(I+1),INTEGER TABLE OUTPUTS
; (N); THRESHOLD, EARLY TERMINATION TEST.
;
; FUNCTION U(K)=SUM J=[0,N-1] OF R(K-J)*Y(J)
; (I)=VZ-1-N,
; (I+1)=E(N)/R(0),RELATIVE ENERGY, "SPIKE SOLUTION"
; IF COMPLETED SUCCESSFULLY, N=VZ-1, I.E. (I)=0
;
; ALGORITHM
; DETERMINES SOLUTION BY FORWARD INTERATION
; MAKES USE OF THE SPIKE CASE, I.E.
; -R(K+1)=SUM FOR J=[0,N-1] OF R(K-J)*A(N-1,J)
;
; INITIALIZATION: E(-1)=R(0)
; RECURSION FORMULA
;
; B(N)=R(N+1)+SUM J=[0,N-1]OF R(N-J)*A(N-1,J)
; C(N)=U(N)-SUM J=[0,N-1] OF R(N-J)*Y(N-1,J)
;
; KB(N), REFLECTANCE,=B(N)/E(N-1)
; KC(N)=C(N)/E(N-1)
; E(N),ENERGY, SPIKE CASE,= E(N-1)-B(N)*KB(N)
;
; Y(N,J)=Y(N-1,J)+KC(N)*A(N-1,N-1-J) ;Y(N,N)=KC(N)
; A(N,J)=A(N-1,J)-KB(N)*A(N-1,N-1-J) ;A(N,N)=-KB(N)
;
; THE EXTERNAL VALUES
; A(N,-1)=1, A(N,J)=Y(N,J)=0, J>N
; ARE CONSISTENT WITH THE RECURSION FORMULA
;
; THE ITERATION NORMALLY TERMINATES WITH N=VZ-1
; EARLY TERMINATION OCCURS IF (N)*MAG [B(N)] > E(N-1)
; IN SUCH CASE UNUSED VALUES OF A,KB, AND Y ARE SET TO ZERO
;
; BUFFER USAGE
; Y(K)=Y(N,K)
; W(K)=A(N,K), 0<=K<N
; W(N+K)=KB(K),  0<=K<N
; NORMAL TERMINATION, N=VZ-1
; EARLY TERMINATION, UNUSED VALUES ZERO FILLED
; (I)=VZ-1-N, N=VZ-1, NORMAL TERMINATION
; (I+1)=E(N)/R(0)=E(N)/E(-1), FRACTIONAL ENERGY
```

```
PAGE   42:  [RNMD]<DCA16>RNN16N.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
             NEWL - APU PROGRAM

                         (01731) * NEWL - APU PROGRAM
                         (01732) ;
                         (01733)         EVEN
        06204 0000       (01734)         DATA NEWL$SA                 ;START ADDRESS
        06205 007E       (01735)         DATA NEWL$SZ                 ;MODULE SIZE
                         (01736) ;
                         (01737) NEWL$B: BEGIN APU(NEWL)
              00000003   (01738)         $M=3
              00000000   (01739)         $A=0
                         (01740) ;
                         (01741) ; REGISTER INITIALIZATION
   A00 06206 00F70000    (01742) NEWL$SA:MOV(IQA,A7) \ NOP          ;A7=YZ, INTEGER
   A01 06208 00F60000    (01743)         MOV(IQA,A6) \ NOP          ;A6=R(0)=E(-1)
                         (01744) ;
                         (01745) ; N=0 ITERATION; B(0) AND C(0) CALCULATION
                         (01746) ;
   A02 0620A 00F50000    (01747)         MOV(IQA,A5) \ NOP          ;A5=B(0)=R(1)
   A03 0620C 02C00000    (01748)         R(A6) \ NOP                ;SET RREG=E(-1)
   A04 0620E 00F40000    (01749)         MOV(IQA,A4) \ NOP          ;A4=C(0)=U(0)
   A05 06210 1000001E    (01750)         JUMP(NEWL4)                ;GO TO RREFLECTANCE CALCULATION
                         (01751) ;
                         (01752) ;
                         (01753) ;
                         (01754) ; B(N) AND C(N) CALCULATION
                         (01755) ; FUNCTION
                         (01756) ; B(N)=R(N+1)+SUMJ=[0,N-1] OF R(N-J)*A(N-1,J)
                         (01757) ; C(N)=U(N)-SUMJ=[0,N-1] OF R(N-J)*Y(N-1,J)
                         (01758) ; SEQUENCING
                         (01759) ; R(N+1),U(N),[A(N-1,J),R(N-J),Y(N-1,J)],NI
                         (01760) ; REGISTER CONTENTS
                         (01761) ; A6=E(N-1), A7=YZ-N
                         (01762) ;
   A06 06212 00F50000    (01763) B1:     MOV(IQA,A5) \ NOP          ;A5=R(N+1), B SUM START
   A07 06214 00F40000    (01764)         MOV(IQA,A4) \ NOP          ;A4=U(N), C SUM START
                         (01765) ;
   A08 06216 00E80000    (01766)         MOV(IQA,M0) \ NOP          ;M0=A(N-1,0)
   A09 06218 00CC0000    (01767)         MOV(IQ,M4) \ NOP           ;M4=R(N)
   A0A 0621A 84000000    (01768)         MUL(M0,M4) \ NOP
                         (01769) ;
   A0B 0621C 00ED0000    (01770)         MOV(IQA,M5) \ NOP          ;A5=R(N)
   A0C 0621E 02000000    (01771)         R(A4) \ NOP                ;SETS R=C SUM
   A0D 06220 00E90000    (01772)         MOV(IQA,M1) \ NOP          ;M1=Y(N-1,0)
   A0E 06222 840F0000    (01773)         MOV(A0),MUL(M1,M5) \ NOP   ;A0=R(N)*A(N-1,0)
   A0F 06224 91160019    (01774)         JUMPS(NEWL3,FWI)           ;EXIT IF DONE
                         (01775) ;
   A10 06226 00E80000    (01776) B2:     MOV(IQA,M0) \ NOP          ;M0=A(N-1,J)
   A11 06228 45140000    (01777)         MOV(A4),ADD(A0,A5) \ NOP   ;A4=CSUM
   A12 0622A 00CC0000    (01778)         MOV(IQ,M4) \ NOP           ;M4=R(N-J)
   A13 0622C 84110000    (01779)         MOV(A1),MUL(M0,M4) \ NOP   ;A1=Y(N-1,J-1)*R(N+1-J)
                         (01780) ;
   A14 0622E 00ED0000    (01781)         MOV(IQA,M5) \ NOP          ;M5=R(N-J)
   A15 06230 49950000    (01782)         MOV(A5),SUB(A4,A1) \ NOP   ;A5=BSUM
   A16 06232 00E90000    (01783)         MOV(IQA,M1) \ NOP          ;A1=Y(N-1,J)
```

```
A17 06234 04800000 (01784)          MOV(A0),MUL(M1,M5) \ NOP    ;A0=A(N-1,J)*R(N-J)
                   (01785) ;
A18 06236 90160010 (01786)          JUMPC(02,FWI)               ;LOOP UNLESS DONE
A19 06238 20372037 (01787) MEWL3:   CLEAR(WI)
                   (01788) ;
A1A 0623A 45140000 (01789)          MOV(A4),ADD(A0,A5) \ NOP    ;A4=CSUM
A1B 0623C 00B10000 (01790)          MOV(P,A1) \ NOP             ;A1=Y(N-1,N-1)*R(1)
A1C 0623E 49950000 (01791)          MOV(A5),SUB(A4,A1) \ NOP    ;A5=B(N)
A1D 06240 02040000 (01792)          MOV(A4),R(A6) \ NOP         ;A4=C(N), SET RREG=E(N-1)
                   (01793) ;
                   (01794) ;
                   (01795)
                   (01796) ;
                   (01797) ; REFLECTANCE CALCULATION AND EARLY TERMINATION TEST
                   (01798) ; FUNCTION
                   (01799) ; IF (B)*ABS B(N) > E(N-1), EARLY TERMINATION, SET AF2
                   (01800) ; A7=A7-1, DECREMENT COUNT
                   (01801) ; R=1/E(N-1)
                   (01802) ;
                   (01803) ; KB(N)=R*B(N), KC(N)=R*C(N),E(N)=E(N-1)-B(N)*KB(N)
                   (01804) ;
                   (01805) ; SEQUENCING
                   (01806) ; INPUT (B),INTEGER
                   (01807) ; OUT KB(N), OUT A(N,N)=-KB(N),OUT Y(N,N)=KC(N)
                   (01808) ;
                   (01809) ; REGISTER CONTENTS
                   (01810) ; A4=C(N), A5=B(N), A6=RREG=E(N-1),A7=YZ-N
                   (01811) ;
                   (01812) ; PART1: RECIPROCAL AND TERMINATION TEST
                   (01813) ;
A1E 06242 2FC80000 (01814) MEWL4:   MOV(M0),RCP(A6) \ NOP       ;M0=E(N-1)
A1F 06244 16AC0000 (01815)          MOV(M4),K(2) \ NOP          ;M4=R0=RCP[E(N-1)]
A20 06246 04000000 (01816)          MUL(M0,M4) \ NOP
                   (01817) ;
A21 06248 02B10000 (01818)          MOV(A1),R(A5) \ NOP         ;A1=2
A22 0624A 028A0000 (01819)          MOV(M2),R(A4) \ NOP         ;M2=B(N)
A23 0624C 00B00000 (01820)          MOV(P,A0) \ NOP             ;A0=R0*RCP[E(N-1)]
A24 0624E 402R0000 (01821)          MOV(M3),SUB(A1,A0) \ NOP    ;M3=C(N)
A25 06250 00890000 (01822)          MOV(R,M1) \ NOP             ;M1=2-R0*RCP[C(N-1)]
                   (01823) ;
A26 06252 04800000 (01824)          MUL(M1,M4) \ NOP
A27 06254 00EF0000 (01825)          MOV(IQA,M7) \ NOP           ;M7=(B),THRESHOLD
A28 06256 00AC0000 (01826)          MOV(P,M4) \ NOP             ;M4=R1=R0*[2-R0*E(N-1)]
A29 06258 04000000 (01827)          MUL(M0,M4) \ NOP
                   (01828) ;
A2A 0625A 00F30000 (01829)          MOV(IQA,A3) \ NOP           ;A3=INTEGER 1
A2B 0625C 4DE00000 (01830)          SUB(A7,A3) \ NOP            ;DECREMENT COUNT
A2C 0625E B5700000 (01831)          MOV(A0),ABSMUL(M2,M7) \ NOP ;A0=R1*E(N-1)
                   (01832) ;
A2D 06260 48370000 (01833)          MOV(A7),SUB(A1,A0) \ NOP    ;A7=YZ-N-1
A2E 06262 00890000 (01834)          MOV(R,M1) \ NOP             ;M1=2-R1*E(N-1)
A2F 06264 049A0000 (01835)          MOV(A0),MUL(M1,M4) \ NOP    ;A0=(B)*B(N)
                   (01836) ;
```

```
A30 06266 76000000 (01837)          MAXABS(A0,A6) \ NOP
A31 06268 00000000 (01838)          MOV(R,NULL) \ NOP           ;TCLEAR IF B*B(N) > E(N-1)
A32 0626A 00AC0000 (01839)          MOV(P,M4) \ NOP             ;M4=R=R1*[2-R1*E(N-1)]
                   (01840) ;
                   (01841) ; PART2: REFLECTANCE CALCULATION
                   (01842) ; ALGORITHM
                   (01843) ; XB(N)=R*B(N), XC(N)=R*C(N)
                   (01844) ; APPROXIMATE VALUES; ONE MORE ITERATION NEEDED
                   (01845) ; KB(N)=XB(N)+R*[B(N)-XB(N)*E(N-1)]
                   (01846) ; KC(N)=XC(N)+R*[C(N)-XC(N)*E(N-1)]
                   (01847) ; E(N)=E(N-1)-B(N)*KB(N)
                   (01848) ;
                   (01849) ; REGISTER CONTENTS
                   (01850) ; M2=A5=B(N); M3=A4=C(N)
                   (01851) ; M0=A6=E(N-1); M4=R,A7=VZ-N-1
                   (01852) ;
A33 0626C 85000000 (01853)          MUL(M2,M4) \ NOP
A34 0626E 911E0036 (01854)          JUMPS(0A+2,T)               ;TCLEAR, FLAG TERMINATE
A35 06270 204A204A (01855)          SET(AF2)
                   (01856) ;
A36 06272 00AE0000 (01857)          MOV(P,M6) \ NOP             ;M6=XB(N)
A37 06274 85900000 (01858)          MOV(A0),MUL(M3,M4) \ NOP    ;A0=XB(N)
                   (01859) ;
A38 06276 00AF0000 (01860)          MOV(P,M7) \ NOP             ;M7=XC(N)
A39 06278 84510000 (01861)          MOV(A1),MUL(M0,M6) \ NOP    ;A1=XC(N)
                   (01862) ;
A3A 0627A 84720000 (01863)          MOV(A2),MUL(M0,M7) \ NOP    ;A2=XB(N)*E(N-1)
                   (01864) ;
A3B 0627C 4AA00000 (01865)          SUB(A5,A2) \ NOP
A3C 0627E 00890000 (01866)          MOV(R,M1) \ NOP             ;M1=B(N)-XB(N)*E(N-1)
A3D 06280 84930000 (01867)          MOV(A3),MUL(M1,M4) \ NOP    ;A3=XC(N)*E(N-1)
                   (01868) ;
A3E 06282 4B800000 (01869)          SUB(A4,A3) \ NOP
A3F 06284 00880000 (01870)          MOV(R,M0) \ NOP             ;M0=C(N)-XC(N)*E(N-1)
A40 06286 84120000 (01871)          MOV(A2),MUL(M0,M4) \ NOP    ;A2=R*[B(N)-XB(N)*E(N-1)]
                   (01872) ;
A41 06288 42000000 (01873)          ADD(A0,A2) \ NOP
A42 0628A 009C0000 (01874)          MOV(R,OQ) \ NOP             ;OQ=KB(N),TO REFLECTANCE LIST
A43 0628C 008C0000 (01875)          MOV(R,M4) \ NOP             ;M4=KB(N)
A44 0628E 85130000 (01876)          MOV(A3),MUL(M2,M4) \ NOP    ;A3=R*[C(N)-XC(N)*E(N-1)]
                   (01877) ;
A45 06290 0A100000 (01878)          MOV(A0),NEG(A0) \ NOP       ;A0=KB(N)
A46 06292 433C0000 (01879)          MOV(OQ),ADD(A1,A3) \ NOP    ;OQ=A(N,N)=-KB(N)
A47 06294 00800000 (01880)          MOV(P,A0) \ NOP             ;A0=B(N)*KB(N)
A48 06296 009C0000 (01881)          MOV(R,OQ) \ NOP             ;OQ=Y(N,N)=KC(N)
A49 06298 48CD0000 (01882)          MOV(M5),SUB(A6,A0) \ NOP    ;M5=KC(N)
A4A 0629A 00960000 (01883)          MOV(R,A6) \ NOP             ;A6=E(N)
                   (01884) ;
A4B 0629C 904A0060 (01885)          JUMPC(NEWL7,AF0),SET        ;SKIP FORWARD ITERATION FOR N=0
                   (01886) ;
                   (01887) ; BASIC FORWARD ITERATION
                   (01888) ;
                   (01889) ; FUNCTION
```

AD-A096 092 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL DCA100-79-C-0037

UNCLASSIFIED BBN-4565-VOL-2 NL

3 OF 5

AD A 096092

```
                             ; A(N,N)=-KB(N); Y(N,N)=KC(N),
                             ; WERE EXECUTED DURING REFLECTANCE CALCULATION
                             ; A(N,J)=A(N-1,J)-KB(N)*A(N-1,N-1-J)
                             ; A(N,N-1-J)=A(N-1,N-1-J)-KB(N)*A(N-1,J)
                             ; Y(N,J)=Y(N-1,J)+KC(N)*A(N-1,N-1-J)
                             ; Y(N,N-1-J)=Y(N,N-1-J)+KC(N)*A(N-1,J)
                             ; AS INDICATED, RECURSION WORKS WOTH ENDS TOWARDS MIDDLE
                             ; FOR N ODD, RECURSION IS DUPLICATED ON J=(N+1)/2 VALUE
                             ;
                             ; SEQUENCING
                             ; A(N-1,N-1),A(N-1,0), JUMP TO FIRST Y INPUT IN LOOP
                             ; [OUT A(N,N-J),A(N-1,N-1-J),OUT Y(N),A(N-1,J),
                             ; OUT Y(N,N-J),Y(N-1,J),OUT A(N,J), Y(N-1,N-1-J)], FWI
                             ; OUT A(N,N-1-N/2), OUT Y(N,N/2), OUT Y(N,N-1-N/2)
                             ;
                             ; REGISTER CONTENTS
                             ; M4=KB(N), M5=KC(N)
                             ; A6=E(N),A7=YCZ-N-1
                             ;
A4C 0629E 08C00000                   MOV(IQ,M0) \ NOP                ;M0=A(N-1,N-1)
A4D 062A0 04000000                   MUL(M0,M4) \ NOP
                             ;
A4E 062A2 08F00000                   MOV(IQA,A0) \ NOP               ;A0=A(N-1,N-1)
A4F 062A4 08C90000                   MOV(IQ,M1) \ NOP                ;M1=A(N-1,0)
A50 062A6 84950000                   MOV(A5),MUL(M1,M4) \ NOP        ;A5=KB(N)*A(N-1,N-1)
                             ;
A51 062A8 08F10000                   MOV(IQA,A1) \ NOP               ;A1=A(N-1,0)
A52 062AA 4D200000                   SUB(A1,A5) \ NOP
A53 062AC 1000005D                   JUMP(MEWL6)
                             ;
A54 062AE 455C0000           05:     MOV(OQ),ADD(A2,A5) \ NOP        ;OQ=A(N,N-J)
A55 062B0 08C80000                   MOV(IQ,M0) \ NOP                ;M0=A(N-1,N-1-J)
A56 062B2 84140000                   MOV(A4),MUL(M0,M4) \ NOP        ;A4=KC(N)*A(N-1,J-1)
                             ;
A57 062B4 08F00000                   MOV(IQA,A0) \ NOP
A58 062B6 447C0000                   MOV(OQ),ADD(A3,A4) \ NOP        ;OQ=Y(N,J)
A59 062B8 08C90000                   MOV(IQ,M1) \ NOP                ;M1=A(N-1,J)
A5A 062BA 84950000                   MOV(A5),MUL(M1,M4) \ NOP        ;A5=KB(N)*A(N-1,N-1-J)
                             ;
A5B 062BC 08F10000                   MOV(IQA,A1) \ NOP               ;A1=A(N-1,J)
A5C 062BE 4D3C0000                   MOV(OQ),SUB(A1,A5) \ NOP        ;OQ=Y(N,N-J)
A5D 062C0 08F20000           MEWL6:  MOV(IQA,A2) \ NOP               ;A2=Y(N-1,J)
A5E 062C2 84340000                   MOV(A4),MUL(M0,M5) \ NOP        ;A4=KB(N)*A(N-1,J)
                             ;
A5F 062C4 4C1C0000                   MOV(OQ),SUB(A0,A4) \ NOP        ;OQ=A(N,J)
A60 062C6 08F30000                   MOV(IQA,A3) \ NOP               ;A3=Y(N-1,N-1-J)
A61 062C8 84B50000                   MOV(A5),MUL(M1,M5) \ NOP        ;A5=KC(N)*A(N-1,N-1-J)
                             ;
A62 062CA 90160054                   JUMPC(05,FWI)
A63 062CC 20372037                   CLEAR(WI)
                             ;
A64 062CE 455C0000                   MOV(OQ),ADD(A2,A5) \ NOP        ;OQ=A(N,N-1-N/2)
A65 062D0 08B40000                   MOV(P,A4) \ NOP
```

```
 A66 062D2 447C0000 (01943)          MOV(DQ),ADD(A3,A4) \ NOP          ;DQ=Y(N,N/2)
 A67 062D4 089C0000 (01944)          MOV(R,DQ) \ NOP                   ;DQ=Y(N,N-1-N/2)
                    (01945) ;
 A68 062D6 900A0006 (01946) MEWL7:   JUMPC(01,AF2)                     ;AF2,CLEAR,GO BACK
                    (01947) ;
                    (01948) ; TERMINATION
                    (01949) ; FUNCTION
                    (01950) ; (I)=VZ-1-N, INTEGER
                    (01951) ; (I+1)=E(N-1)/R(0), FRACTION, FIXED
                    (01952) ; Y,A,KB,ZERO FILLED IF NECESSARY
                    (01953) ; SEQUENCING
                    (01954) ; R(0),INTEGER1,OUT(I), OUT(I+1),
                    (01955) ; OUT Y,A,KB,AS NECESSARY
                    (01956) ; REGISTER CONTENTS
                    (01957) ; A6=E(N-1), A7=VZ-2-N
                    (01958) ;
                    (01959) ;
 A69 062D8 08D00000 (01960)          MOV(IQ,A0) \ NOP                  ;A0=R(0)
 A6A 062DA 2E000000 (01961)          RCP(A0) \ NOP
 A6B 062DC 08E80000 (01962)          MOV(IQA,M0) \ NOP                 ;M0=R(0)
 A6C 062DE 16AC0000 (01963)          MOV(M4),K(2) \ NOP                ;M4=RCP(A0)
 A6D 062E0 84000000 (01964)          MUL(M0,M4) \ NOP
                    (01965) ;
 A6E 062E2 08B70000 (01966)          MOV(P,A2) \ NOP                   ;A2=R(0)*RCPCR(0)3
 A6F 062E4 4A310000 (01967)          MOV(A1),SUB(A1,A2) \ NOP          ;A1=2
 A70 062E6 3AE90000 (01968)          MOV(M1),ALIGN(A7) \ NOP           ;M1=2-R(0)*RCPCR(0)3
 A71 062E8 84800000 (01969)          MUL(M1,M4) \ NOP
                    (01970) ;
 A72 062EA 02DC0000 (01971)          MOV(DQ),R(A6) \ NOP               ;DQ=VZ-1-N, INTEGER
 A73 062EC 088D0000 (01972)          MOV(R,M5) \ NOP                   ;M5=E(N-1)
 A74 062EE 08A90000 (01973)          MOV(P,M1) \ NOP                   ;M1=1/R(0)
 A75 062F0 84A00000 (01974)          MUL(M1,M5) \ NOP
                    (01975) ;
 A76 062F2 08B00000 (01976)          MOV(P,A0) \ NOP                   ;A0=E(N-1)/R(0)
 A77 062F4 3A000000 (01977)          ALIGN(A0) \ NOP
 A78 062F6 089C0000 (01978)          MOV(R,DQ) \ NOP                   ;DQ=E(N-1)/R(0),FIXED
                    (01979) ;
 A79 062F8 081C0000 (01980)          MOV(ZERO,DQ) \ NOP                ;ZERO FILL
 A7A 062FA 901D0079 (01981)          JUMPC(0A-1,FI)
 A7B 062FC 20322032 (01982)          CLEAR(RA)
 A7C 062FE 00000000 (01983)          NOP
 A7D 06300 10000000 (01984)          JUMP(MEWL$SA)
                    (01985) ;
          0000007E  (01986) MEWL$SZ=0A-MEWL$SA
     06302          (01987)          END
                    (01988) ;
                    (01989) ;
```

```
                     (01990) ' MEWL - APS PROGRAM
                     (01991) ; INPUT ONLY MODE
                     (01992) ; SEQUENCING LISTED WITH APU PROGRAM
                     (01993) ;
                     (01994) ; BINDING PARAMETERS
                     (01995) ;
   06302 0010        (01996) YB      DATA $10              ;YBASE, MAY BE IN PLACE, I.E. Y=U
                     (01997) ;        YS=2                 ;YSEPARATION; MUST BE COMPACT 32 BIT
                     (01998) ;YZ      DATA 4               ;YSIZE DETERMINES SIZE
   06303 0002        (01999) YBUS    DATA 2                ;Y BUS
                     (02000) ;
   06304 0020        (02001) UB      DATA $20              ;UBASE
   06305 0002        (02002) UBUS    DATA 2                ;UBUS
                     (02003) ;        US=2                 ;USEPARATION, MUST BE COMPACT 32 BIT
                     (02004) ;        UZ=YZ                ;USIZE, NOT USED BY PROGRAM
                     (02005) ;
   06306 0030        (02006) VB      DATA $30              ;VBASE, V(K)=R(K)=R(-K)
   06307 0002        (02007) VBUS    DATA 2                ;V BUS
                     (02008) ;        VS=2                 ;VSEPARATION, MUST BE COMPACT 32 BIT
                     (02009) ;        VZ>YZ                ;VSIZE, NOT USED BY PROGRAM
                     (02010) ;
   06308 0040        (02011) WB      DATA $40              ;WBASE
   06309 0002        (02012) WBUS    DATA 2                ;W BUS
                     (02013) ;        WS=2                 ;W MUST BE COMPACT 32 BIT
                     (02014) ;        WZ=2*YZ              ;W(K)=A(K), K<N
                     (02015) ;                             ;W(N+K)=KB(K), K<N
                     (02016) ;
   0630A 0000        (02017) ITI      DATA $0              ;LOCATION, INTEGER OUTPUT
   0630B 0002        (02018) IBUS     DATA 2
   0630C 0002        (02019) SB       DATA $2              ;LOCATION, SCALAR B
   0630D 0002        (02020) SBUS     DATA 2
                     (02021) ;
                     (02022) ;
                     (02023)          EVEN
   0630E 00000000    (02024)          ADDR 0               ;PTR TO CONS INS BLOCK
   06310 00000000    (02025)          ADDR 0               ;SCALAR ADDRESS
   06312 0000        (02026)          DATA 0               ;NUMBER OF SCALARS
   06313 00A6        (02027)          DATA MEWL$Z          ;MODULE SIZE
   06314 00000000    (02028)          ADDR MEWL$C          ;PTR TO CHAIN ANCHOR
                     (02029) ;
                     (02030) MEWL$:  BEGIN APS(MEWL)
                     (02031) ;
                     (02032) ; BINDING DONE BY SBM$MEWL
                     (02033) ;
                     (02034)          OPADD WAFNE, (3 .LS. 10)+(12 .LS. 5)+X'1A'
         00000000    (02035)          $A=0
                     (02036) ;
                     (02037) ; REGISTER INITIALIZATION
                     (02038) ;
A00 06316 0142631B   (02039)          LOAD(BR0,MYZ(1),L,TE)      ;INTEGER YZ, INPUT
                     (02040) ;
A01 06318 02300032   (02041)          SET(RA)                    ;TURN ON AP
         0000631B    (02042) MYZ=$L+1
```

```
PAGE   40:  [BBND]<DCA16>BBN16M.MSU.2, 29-Dec-80 14:30:40, Ed: WOLF
                MEWL - APS PROGRAM

  A02 0631A 0440000F (02043) MEWLA1: LOAD(BR0,YZ)
  A03 0631C 06400000 (02044)         LOAD(BR3,0)
  A04 0631E 00310011 (02045)         MOVB(BW3,BR0)           ;BW3=2*N=0,STARTING
                     (02046) ;
  A05 06320 0AC46306 (02047) MEWLA2: LOAD(BR0,VR(2),TF)      ;BR0=VBASE=R(0),INPUT
  A06 06322 0C646308 (02048) MEWLA3: LOAD(BR2,WB(2),L)       ;BR2=WBASE
  A07 06324 0E2A001E (02049) MEWLA4: ADD(BR2,2*YZ)           ;BR2=KB(0)
  A08 06326 10220002 (02050)         SUB(BR2,2)              ;BR2=KB(-1)
  A09 06328 12210015 (02051)         MOVB(BW2,BR2)           ;BW2=KB(-1)
                     (02052) ;
                     (02053) ; B(N) AND C(N) CALCULATION
                     (02054) ; REGISTER CONTENTS, BW2=KB(N-1), BW3=2*N
                     (02055) ;
  A0A 0632A 14446308 (02056) MEWLA5: LOAD(BR0,WB(2),L)       ;BR0=WBASE
  A0B 0632C 16020002 (02057)         SUB(BR0,2)              ;BR0=A(-1)
  A0C 0632E 18546306 (02058) MEWLA6: LOAD(BR1,VB(2),L)       ;BR1=R(0)
  A0D 06330 1A646302 (02059) MEWLA7: LOAD(BR2,YB(2),L)       ;BR2=Y(0)
  A0E 06332 1C220002 (02060)         SUB(BR2,2)              ;BR2=Y(-1)
  A0F 06334 1E0043EA (02061)         JUMPS(MEWLA16,AF2)      ;EXIT IF N=YZ
                     (02062) ;
  A10 06336 2019002E (02063)         ADDB(BR1,BW3)           ;BR1=R(N)
  A11 06338 229A0002 (02064)         ADD(BR1,2,TF)           ;BR1=R(N+1), INPUT
  A12 0633A 24746304 (02065) MEWLA8: LOAD(BR3,UB(2),L)       ;BR3=U(0)
  A13 0633C 26B9002E (02066)         ADDB(BR3,BW3,TF)        ;BR3=U(N),INPUT
                     (02067) ;
  A14 0633E 28001DA0 (02068)         JUMPC(MEWLA11,AF3)      ;OMIT DOT PROD FIRST PASS
  A15 06340 2A0015BA (02069)         JUMPC(8A,WAFNE)         ;WAIT ON WAF EMPTY
                     (02070) ;
  A16 06342 2C111A56 (02071)         MOVB(BW1,BW3),JUMP(MEWLA10)  ;BW1=2*N, COUNTER
                     (02072) ;
  A17 06344 2E0A0002 (02073) 89:     ADD(BR0,2,TF)           ;BR0=A(J),INPUT
  A18 06346 30920002 (02074)         SUB(BR1,2,TF)           ;BR1=R(N-J), INPUT
  A19 06348 32AA0002 (02075)         ADD(BR2,2,TF)           ;BR2=Y(J), INPUT
  A1A 0634A 341117B2 (02076) MEWLA10:SUBL(BW1,2),JUMPP(89)
                     (02077) ;
  A1B 0634C 36300037 (02078)         SET(WI)
  A1C 0634E 30000020 (02079)         NOP
                     (02080) ;
                     (02081)         EJECT
```

```
                        (02082) ; REFLECTANCE CALCULATION AND TERMINATION TEST
                        (02083) ; REGISTER CONTENTS
                        (02084) ; BR0=A(N-1), BR1=R(1), BR2=Y(N-1)
                        (02085) ; BW2=KB(N-1), BW3=2*N
                        (02086) ;
A1D 06350 3AF2630C (02087) NEWLA11:LOAD(BR3,SB(1),TF)          ;INPUT(B),THRESHOLD
A1E 06352 3D72630C (02088) NEWLA12:LOAD(BR3,NEWLI1(1),TE)      ;INTEGERI INPUT
                        (02089) ;
A1F 06354 3E390016 (02090)         MOVB(BR3,BW3)               ;BW3=2*N
A20 06356 4032001E (02091) NEWLA13:SUB(BR3,2*YZ)               ;BR3=2*(N-YZ)
A21 06358 423923FA (02092)         ADDL(BR3,2),JUMPN(0A+2)     ;BR3=2*(N+1-YZ)
A22 0635A 4430002A (02093)         SET(AF2)                    ;AF2 SET, LAST PASS
                        (02094) ;
A23 0635C 46390015 (02095)         MOVB(BR3,BR2)               ;BR3=Y(N-1)
A24 0635E 48290026 (02096)         SUBB(BR2,BW3)               ;BR2=Y(-1)
A25 06360 4A090026 (02097)         SUBB(BR0,BW3)               ;BR0=A(-1)
A26 06362 4C120002 (02098)         SUB(BR1,2)                  ;BR1=R(0)
                        (02099) ;
A27 06364 4E190011 (02100)         MOVB(BR1,BR0)               ;BR1=A(-1)
A28 06366 5019002E (02101)         ADDB(BR1,BW3)               ;BR1=A(N-1)
A29 06368 521A0002 (02102)         ADD(BR1,2)                  ;BR1=A(N)
A2A 0636A 543A0002 (02103)         ADD(BR3,2)                  ;BR3=Y(N)
A2B 0636C 56110016 (02104)         MOVB(BW1,BW3)               ;BW1=2*N
A2C 0636E 5831003A (02105)         ADDL(BW3,2)                 ;BW3=2*(N+1), INCREMENT
                        (02106) ;
A2D 06370 5AA1003A (02107)         ADDL(BW2,2,TF)              ;BW2=KB(N), OUTPUT
A2E 06372 5C010013 (02108)         MOVB(BW0,BR1,TF)            ;BW0=A(N), OUTPUT
A2F 06374 5E010017 (02109)         MOVB(BW0,BR3,TF)            ;BW0=Y(N), OUTPUT
                        (02110) ;
                        (02111) ; ITERATION ROUTINE, BOTH ENDS TOWARDS MIDDLE
                        (02112) ; REGISTER CONTENTS
                        (02113) ; BR0=A(-1), BR1=A(N), BR2=Y(-1), BR3=Y(N)
                        (02114) ; BW1=2*N, BW2=KB(N), BW3=2*(N+1)
                        (02115) ; NOTE RECURSION OMITTED FIRST PASS
                        (02116) ;
A30 06376 6030 0AAB (02117)         JUMPC(NEWLA5,AF3),SET       ;OMIT RECURSION FIRST PASS
A31 06378 62920002 (02118)         SUB(BR1,2,TF)                ;BR1=A(N-1), INPUT
A32 0637A 640A0002 (02119)         ADD(BW0,2,TF)                ;BR0=A(0), INPUT
A33 0637C 66113972 (02120)         SUBL(BW1,2),JUMP(NEWLA15)    ;BW1=2*(N-1), CTR
                        (02121) ;
A34 0637E 68810013 (02122) 014:    MOVB(BW0,BR1,TF)            ;BW0=A(N-J), OUTPUT
A35 06380 6A920002 (02123)         SUB(BR1,2,TF)               ;BR1=A(N-1-J), INPUT
A36 06382 6C810015 (02124)         MOVB(BW0,BR2,TF)            ;BW0=Y(J-1), OUTPUT
A37 06384 6F8A0002 (02125)         ADD(BR0,2,TF)               ;BR0=A(J), INPUT
A38 06386 70810017 (02126)         MOVB(BW0,BR3,TF)            ;BW0=Y(N-J), OUTPUT
                        (02127) ;
A39 06388 72AA0002 (02128) NEWLA15:ADD(BR2,2,TF)               ;BR2=Y(J), INPUT
A3A 0638A 74810011 (02129)         MOVB(BW0,BR0,TF)            ;BW0=A(J), OUTPUT
A3B 0638C 76B20002 (02130)         SUB(BR3,2,TF)               ;BR3=Y(N-1-J), INPUT
                        (02131) ;
A3C 0638E 781134B4 (02132)         SUBL(BW1,4),JUMPP(014)
A3D 06390 7A300037 (02133)         SET(VI)
A3E 06392 7C000020 (02134)         NOP
```

```
                      (Ø2135) ;
A3F Ø6394 7EØ1ØØ13 (Ø2136)        MOVB(BWØ,BR1,TF)            ;BWØ=A(N-N/2), OUTPUT
A4Ø Ø6396 8Ø81ØØ15 (Ø2137)        MOVB(BWØ,BR2,TF)            ;BWØ=Y(N/2-1), OUTPUT
A41 Ø6398 82Ø1ØØ17 (Ø2138)        MOVB(BWØ,BR3,TF)            ;BWØ=Y(N-N/2), OUTPUT
                      (Ø2139) ;
A42 Ø639A 84ØØØA6Ø (Ø214Ø)        JUMP(NEWLA5)
                      (Ø2141) ;
                      (Ø2142) ; TERMINATION ROUTINE
                      (Ø2143) ; REGISTER CONTENTS
                      (Ø2144) ; BRØ=A(-1), BR1=R(Ø), BR2=Y(-1), B
                      (Ø2145) ; BW2=KB(N-1), BW3=2*(N+1)
                      (Ø2146) ; ENTRY UNDER TWO CASES
                      (Ø2147) ; N=YZ-1, COMPLETION,
                      (Ø2148) ; N<YZ-1, EARLY TERMINATION DUE TO KB(N) > 1(B)
                      (Ø2149) ; THE LATTER CASE REQUIRING ZERO FILLING
                      (Ø215Ø) ;
A43 Ø639C Ø69AØØØØ (Ø2151) NEWLA16:ADD(BR1,Ø,TF)          ;BRØ=R(Ø), INPUT
A44 Ø639E Ø8526301 (Ø2152)        LOAD(BR1,ITI(1),L)          ;BR1=I ADDRESS
A45 Ø63AØ Ø8Ø1ØØ13 (Ø2153)        MOVB(BWØ,BR1,TE)            ;OUTPUT (I)=YZ-1-N
                      (Ø2154) ;
A46 Ø63A2 ØC7ØØØ1E (Ø2155) NEWLA17:LOAD(BR3,2*YZ)         ;BR3=2*YZ
A47 Ø63A4 ØE39ØØ26 (Ø2156)        SUBB(BR3,BW3)               ;BR3=2*(YZ-N-1)
                      (Ø2157) ;
                      (Ø2158) ; NOTE (YZ-1-N) ARE # OF DATA POINTS TO ZERO FILL
                      (Ø2159) ;
A48 Ø63A6 9ØØ9ØØ2E (Ø216Ø)        ADDB(BRØ,BW3)               ;BRØ=A(N-1)
A49 Ø63A8 9229ØØ2E (Ø2161)        ADDB(BR2,BW3)               ;BR2=Y(N-1)
A4A Ø63AA 9411ØØ11 (Ø2162)        MOVB(BW1,BRØ)               ;BW1=A(N-1)
A4B Ø63AC 9631ØØ15 (Ø2163)        MOVB(BW3,BR2)               ;BW3=Y(N-1)
                      (Ø2164) ;
A4C Ø63AE 99Ø15Ø79 (Ø2165)        ADDL(BWØ,1,TE),JUMP(NEWLA19)     ;OUT (I+1)=E(N)/R(Ø)
                      (Ø2166) ;
A4D Ø63BØ 9A91ØØ3A (Ø2167) $1Ø:   ADDL(BW1,2,TF)              ;ZERO FILL A(N-1+J)
A4E Ø63B2 9CA1ØØ3A (Ø2168)        ADDL(BW2,2,TF)              ;ZERO FILL KB(N-1+J)
A4F Ø63B4 9EB1ØØ3A (Ø2169)        ADDL(BW3,2,TF)              ;ZERO FILL Y(N-1+J)
A5Ø Ø63B6 AØ394DB2 (Ø217Ø) NEWLA19:SUBL(BR3,2),JUMPP($1Ø)
                      (Ø2171) ;
A51 Ø63B8 A22ØØØ31 (Ø2172)        CLEAR(RI)
A52 Ø63BA A4ØØØØ2Ø (Ø2173)        NOP(Ø)
                      (Ø2174) ;
          ØØØØØØØØ (Ø2175) NEWL$C=ØC
    Ø63BC          (Ø2176)        END
          ØØØØØØA6 (Ø2177) NEWL$Z=ØL-NEWL$
          ØØØØ638C (Ø2178)        NEWLI1=ØL
    Ø63BC ØØØ1     (Ø2179)        DATA $ØØØ1
```

```
                    (02180) ' SBM$MEWL - SPECIAL SUPPORT ROUTINE FOR MEWL
                    (02181) ;
                    (02182) ;
063BD C63400E0 (02183) SBM$MEWL PUSHMIL R3,APDT$ORG(R2)      STACK APU MODULE BUS ORIGIN
063BF 9071FFFE (02184)          MOVIR R7,-W$
063C1 C607FFFE (02185)          PUSHMIL R3,0-W$(R3)          STACK APU MODULE ST ADDR AND SZ
063C3 C63400EA (02186)          PUSHMIL R3,APDT$ORG+W$(R2)   STACK APS MODULE BUS ORIGIN
063C5 C63400EC (02187)          PUSHMIL R3,APDT$ORG+2*W$(R2) STACK CSPU SUPPORT MODULE
063C7 2771     (02188)          DECR R7,1
               (02189)          EVEN
063C8 C407FFFC (02190)          PUSHMI R3,0-2*W$(R3)         APS MODULE SIZE
063CA 9047FFFD (02191)          MOVIR R4,-3*W$(R3)           GET SPECIAL SUPPORT SEMAPHORE
063CC 7040FF00 (02192)          MOVWK R4,R4,MSK$LBYT
063CE 3C40     (02193)          LRS R4,8
063CF 3430     (02194)          PUSHXI R3,R4                 STACK SPECIAL SUPPORT SEMAPHORE
063D0 4057     (02195)          MOVRR R5,R1                  POINT TO FIRST SCALAR ID
063D1 2651     (02196)          INCR R5,W$
063D2 704100FF (02197)          MOVWK R4,R5,MSK$RBYT         EXTRACT SCALAR A IDENTIFIER
063D4 3630     (02198)          PUSHXIL R3,R4                STACK SCALAR A IDENTIFIER
063D5 C6300792 (02199)          PUSHMIL R3,CLR$G0G1          RESET G FLAGS
063D7 2632     (02200)          INCR R3,2
063D8 C63007BA (02201)          PUSHMIL R3,ZERO              SET FOR NO PBF
063DA 9E300011 (02202)          SUBIR R3,AP$LLOC
               (02203) ;
               (02204) ;
               (02205) ; SCALAR BINDING
063DC F0420002 (02206)          MOVMR R4,2*W$(R1)            ;GET SCALAR B ID
063DE 914000FF (02207)          ANDIR R4,MSK$RBYT
063E0 3A41     (02208)          LLS R4,1
063E1 9C400302 (02209)          ADDIR R4,SVT$
063E3 E0406351 (02210)          MOVRM R4,MEWL$+W$*MEWLA11+W$ ;STORED APS INSTR IN MAIN MEM
               (02211) ;
               (02212) ; INTEGER TABLE BINDING (SCALAR A)
063E5 F0420001 (02213)          MOVMR R4,W$(R1)              ;SCALAR A ID
063F7 9A4000FF (02214)          ANDIR R4,MSK$RBYT
063F9 9C400502 (02215)          ADDIR R4,ISVT$
063EB E040639F (02216)          MOVRM R4,MEWL$+W$*MEWLA16+3*W$  ;STORED APS INSTR IN MAIN MEM
               (02217) ;
               (02218) ; YBASE BINDING
063ED F0620001 (02219)          MOVMR R6,W$(R1)
063EF 5060FF00 (02220)          MOVKR R6,R6,MSK$LBYT
063F1 3C67     (02221)          LRS R6,7
               (02222)          EVEN
063F2 C04C0582 (02223)          MOVMRL R4,BCT$BA(R6)         ;GET 17-BIT YBASE ADDR
063F4 F0706330 (02224)          MOVMR R7,MEWL$+W$*MEWLA7
063F6 564FFFF0 (02225)          IORKR R4,R7,$FFF0
063F8 84406330 (02226)          MOVRML R4,MEWL$+W$*MEWLA7
               (02227) ;
               (02228) ;  YZ BINDING
063FA F04C0605 (02229)          MOVMR R4,W$+BCT$AD(R6)       ;R4=BUFFER SIZE-1
063FC 2641     (02230)          INCR R4,1                    ;R4=BUFFER SIZE
063FD E040631B (02231)          MOVRM R4,MEWL$+W$*MEWLA1+W$
063FF 3A41     (02232)          LLS R4,1                     ;R4=R4*2
```

```
06400 E0406325 (02233)          MOVRN R4,MEWL$+W$*MEWLA4+W$
06402 E0406357 (02234)          MOVRN R4,MEWL$+W$*MEWLA13+W$
06404 E04063A3 (02235)          MOVRN R4,MEWL$+W$*MEWLA17+W$
               (02236) ;
06406 F0620002 (02237)          MOVMR R6,2*W$(R1)             ;GET U ID
06408 506CFF00 (02238)          MOVKR R6,R6,MSK$LBYT
0640A 3C67     (02239)          LRS R6,7
0640B 0000     (02240)          EVEN
0640C C04C0582 (02241)          MOVMRL R4,BCT$BA(R6)          ;GET 17-BIT UBASE ADDR
0640E F070633A (02242)          MOVMR R7,MEWL$+W$*MEWLA8
06410 564EFFF0 (02243)          IORKR R4,R7,$FFF0
06412 8440633A (02244)          MOVRML R4,MEWL$+W$*MEWLA8
               (02245) ;
               (02246) ; VBASE BINDING
06414 F0620003 (02247)          MOVMR R6,3*W$(R1)             ;GET VBASE ID
06416 506CFF00 (02248)          MOVKR R6,R6,MSK$LBYT
06418 3C67     (02249)          LRS R6,7
06419 0000     (02250)          EVEN
0641A C04C0582 (02251)          MOVMRL R4,BCT$BA(R6)
0641C F0706320 (02252)          MOVMR R7,MEWL$+W$*MEWLA2
0641E 564EFFF0 (02253)          IORKR R4,R7,$FFF0
06420 84406320 (02254)          MOVRML R4,MEWL$+W$*MEWLA2
               (02255) ;
06422 C04C0582 (02256)          MOVMRL R4,BCT$BA(R6)
06424 F070632E (02257)          MOVMR R7,MEWL$+W$*MEWLA6
06426 564EFFF0 (02258)          IORKR R4,R7,$FFF0
06428 8440632E (02259)          MOVRML R4,MEWL$+W$*MEWLA6
               (02260) ;
               (02261) ; WBASE BINDING
0642A F0620004 (02262)          MOVMR R6,4*W$(R1)             ;GET W ID
0642C 506CFF00 (02263)          MOVKR R6,R6,MSK$LBYT          ;GET PROPER BYTE
0642E 3C67     (02264)          LRS R6,7
0642F 0000     (02265)          EVEN
06430 C04C0582 (02266)          MOVMRL R4,BCT$BA(R6)          ;GET 17-BIT WBASE
06432 F0706322 (02267)          MOVMR R7,MEWL$+W$*MEWLA3
06434 564EFFF0 (02268)          IORKR R4,R7,$FFF0
06436 84406322 (02269)          MOVRML R4,MEWL$+W$*MEWLA3
               (02270) ;
06438 C04C0582 (02271)          MOVMRL R4,BCT$BA(R6)
0643A F070632A (02272)          MOVMR R7,MEWL$+W$*MEWLA5
0643C 564EFFF0 (02273)          IORKR R4,R7,$FFF0
0643E 8440632A (02274)          MOVRML R4,MEWL$+W$*MEWLA5
               (02275) ;
               (02276) ;
               (02277) ;
               (02278) ;
               (02279) ;
               (02280) ;
               (02281) ;
               (02282) ;
06440 80000F63 (02283)         JMP AP$BNDR0
```

```
" APU3 - STAB(V,A,U,V)
* WHR 8/80
*
*  ROUTINE TO CHECK STABILITY OF
*  PITCH PREDICTOR COEFFICIENTS
*  ALSO CODES AND QUANTIZES COEFF'S
*
*  THE INPUT COEFFICIENTS ARE : V(0),V(1),V(2)
*  THEY ARE LINEARLY TRANSFORMED AS:
*
*  T1=V(0)+V(1)+V(2)
*  T2=V(0)-2*V(1)+V(2)
*  T3=V(0)-V(2)
*
*  THEN  ,  IF (T1+1<0 .OR.
*                T2+1<0 .OR.
*                T3+1<0 .OR.
*                T2-2>0 .OR.
*                T3-1>0  )
*  THE FILTER IS DECLARED UNSTABLE
*  AND THE ONE-TAP FILTER IS USED AS
*       C1=0.
*       C2=SA , SA=ONE-TAP COEFF
*       C3=0.
*  OTHERWISE THE
*  FILTER IS DECLARED STABLE AND
*       C1=V(0)
*       C2=V(1)
*       C3=V(2)
*
*  CODE AND QUANTIZE C1,C2,C3
*  SET  V=BUFFER OF QUANTIZED VALUES
*       U=BUFFER OF CODED VALUES
*
*  APU REGISTER USAGE
*       LEFT            RIGHT
*  A0   V(0):C1         V(0):C1
*  A1   V(1):C2         V(1):C2
*  A2   V(2):C3         V(2):C3
*  A3   T1              T2
*  A4   T3
*  A5   +1:C1,C2,C3     +1:-2:CODE
*  A6   CTHC*(N)        2**-15
*  A7   SA:DVLC*(N)     SA:
*
*  INPUT AND OUTPUT SEQUENCE
*  IQ:  V(0),V(1),V(2),SA,2**-15
*       ,(CTHC*(N),DVLC*(N), N=1,...,LENGTH(WI))
*
*  UQ:  (FLT PT QUANTIZED VALUE(I),FXD PT CODE(I),I=1,2,3)
*
*
*  STAB-APU INITIALIZATION
```

```
                      (02337) *
                      (02338) *
                      (02339)         EVEN                        ;START ON WORD BOUNDRY
     06442 000F       (02340)         DATA    STAB$SA             ;START ADDRESS AT SA
     06443 0033       (02341)         DATA    STAB$SZ             ;SIZE , COMPUTED AT END
                      (02342) *
                      (02343) *
                      (02344) STAB$:  BEGIN APU(STAB)             ;REFERENCE BEGIN STATMENT
           00000000   (02345)         STAB$SA=$A
                      (02346) *
                      (02347) *  INPUT COEFFS V(0) AND V(1)
A00 06444 00F000F0    (02348)         MOV(IQA,A0)                 ;A0=V(0)
A01 06446 00F100F1    (02349)         MOV(IQA,A1)                 ;A1=V(1)
                      (02350) *  START TRANSFORM
A02 06448 41004900    (02351)         ADD(A0,A1)\SUB(A0,A1)       ;R=V(0)+V(1) \ R=V(0)-V(1)
                      (02352) *  INPUT COEFF V(2)
A03 0644A 00F200F2    (02353)         MOV(IQA,A2)                 ;A2=V(2)
A04 0644C 42734973    (02354)         MOV(A3),ADD(A3,A2)\MOV(A3),SUB(A3,A1)
                      (02355) *                                   ;A3=V(0)+V(1),R=V(0)+V(1)+V(2)
                      (02356) *                                   ;\A3=V(0)-V(1),R=V(0)-2*V(1)
A05 0644E 4A134273    (02357)         MOV(A3),SUB(A0,A2)\MOV(A3),ADD(A3,A2)
                      (02358) *                                   ;A3=T1,R=V(0)-V(2)
                      (02359) *                                   ;\A3=V(0)-2*V(1),R=T2=V(0)-2V(1)+V(2)
A06 06450 00940093    (02360)         MOV(R,A4)\MOV(R,A3)         ;A4=T3=V(0)-V(2)
                      (02361) *                                   ;\A3=T2=V(0)-2V(1)+V(2)
                      (02362) *  TRANSFORM COMPLETE
                      (02363) *  INPUT 1-TAP COEFF
A07 06452 00F700F7    (02364)         MOV(IQA,A7)                 ;A7=SA
                      (02365) *  START STABILITY CHECK
A08 06454 16001600    (02366)         K(+1)                       ;R=CONSTANT=+1
A09 06456 00950095    (02367)         MOV(R,A5)                   ;A5=R=+1
A0A 06458 45604560    (02368)         ADD(A3,A5)\ADD(A3,A5)       ;T1+1\T2+1
A0B 0645A 00800080    (02369)         MOV(R,NULL)                 ;ENSURE TB IS SET
A0C 0645C 911F0017    (02370)         JUMPS(STAB$UN,T1)           ;JUMP IF TB1 SET , T1+1=R<0
                      (02371) *                                   ; T1<-1
A0D 0645E 911F0017    (02372)         JUMPS(STAB$UN,T2)           ;JUMP IF TB2 SET , T2+1=R<0
                      (02373) *                                   ; T2<-1
A0E 06460 000006A0    (02374)         NOP\ K(-2)                  ;NOP \ R=-2
A0F 06462 00000095    (02375)         NOP\ MOV(R,A5)              ;NOP \ A5=-2
A10 06464 45804560    (02376)         ADD(A4,A5)\ADD(A3,A5)       ;R=T3+1 \ R=T2-2
A11 06466 00800080    (02377)         MOV(R,NULL)                 ;ENSURE TB IS SET
A12 06468 911F0017    (02378)         JUMPS(STAB$UN,T1)           ;JUMP IF TB1 SET , T3+1=R<0
                      (02379) *                                   ;T3<-1
A13 0646A 901F0017    (02380)         JUMPC(STAB$UN,T2)           ;JUMP IF TB2 IS CLEAR , T2-2=R>0
                      (02381) *                                   ;T2>2
A14 0646C 4D800000    (02382)         SUB(A4,A5)\NOP              ;R=T3-1\NOP
A15 0646E 00800000    (02383)         MOV(R,NULL)\NOP             ;ENSURE TB IS SET
A16 06470 911E001B    (02384)         JUMPS(STAB$QT,T1)           ;JUMP IF TB1 IS SET TO QUANTIZATION
                      (02385) *  3-TAP FILTER UNSTABLE - REPLACE
                      (02386) *  WITH ONE TAP FILTER.
                      (02387) STAB$UN:
A17 06472 02E002E0    (02388)         R(A7)                       ;R=SA
A18 06474 00100010    (02389)         MOV(ZERO,A0)                ;A0=C1=0.0
```

```
A19 06476 00120012 (02390)          MOV(ZERO,A2)              ;A2=C3=0.0
A1A 06478 00910091 (02391)          MOV(R,A1)                 ;A1=C2=SA=1-TAP COEFF
                   (02392) *
                   (02393) *START OF QUANTIZATION / CODING
A1B 0647A 000000F6 (02394) STAB$QT: NOP\MOV(IQA,A6)           ;NOP\2**-15 FOR FIXED PT COUNTING
                   (02395) *  CALL SUBROUTINE CQ$SB  FOR EACH COEFF
                   (02396) *  1ST TAP INPUT
A1C 0647C 02000000 (02397)          R(A0)\NOP                 ;R=A0=C1L\NOP
A1D 0647E 30000024 (02398)          CALL(CQ$SB)               ;CALL QUANTIZATION SUBROUTINE
                   (02399) *  2ND TAP INPUT
A1E 06480 02200000 (02400)          R(A1)\NOP                 ;R=A1=C2 \ NOP
A1F 06482 30000024 (02401)          CALL(CQ$SB)               ;CALL QUANTIZATION SUBROUTINE
                   (02402) *  3RD TAP INPUT
A20 06484 02400000 (02403)          R(A2)\NOP                 ;R=A2=C3 \ NOP
A21 06486 30000024 (02404)          CALL(CQ$SB)               ;CALL QUANTIZATION SUBROUTINE
                   (02405) *
A22 06488 20322032 (02406)          CLEAR(RA)                 ;HALT APU
A23 0648A 10000000 (02407)          JUMP(0)
                   (02408) *
                   (02409) *  *  *  CQ$SB - QUANTIZATION SUBROUTINE
                   (02410) *
                   (02411) *  PUT COEFF IN A5 AND INIT CODE
A24 0648C 00950015 (02412) CQ$SB:  MOV(R,A5)\MOV(ZERO,A5)    ;INPUT COEFF \ INIT CODE=0
A25 0648E 000002A0 (02413)          NOP\R(A5)                 ;NOP\R=A5=ZERO
                   (02414) *  INNER LOOP - FIND QUANT VALUE AND CODE
A26 06490 00F60000 (02415) 06:      MOV(IQA,A6)\NOP           ;INPUT BOUNDRY - CTH(N)\NOP
A27 06492 4EA04685 (02416)          SUB(A5,A6)\MOV(A5),ADD(A5,A6)
                   (02417) *                                  ;(C-CTH(N))\UPDATE CODE,CODE+1
A28 06494 00F70000 (02418)          MOV(IQA,A7)\NOP           ;INPUT QUANTIZED VALUE OVL(N)\NOP
A29 06496 00800000 (02419)          MOV(R,NULL)\NOP           ;ENSURE TB1 SET
A2A 06498 9116002F (02420)          JUMPS(STAB$OT,FWI)        ;JUMP IF DONE TABLE
A2B 0649A 901F0026 (02421)          JUMPC(06,T1)              ;JUMP IF C > CTH (KEEP LOOKING)
                   (02422) *  FOUND THE RIGHT QUANTIZATION BIN. READ IN REST OF TABLE.
A2C 0649C 00E000E0 (02423) 07:      MOV(IQA,NULL)             ;INPUT REST OF TABLE
A2D 0649E 00000000 (02424)          NOP                       ;TO ALLOW FWI TO SET
A2E 064A0 9016002C (02425)          JUMPC(07,FWI)             ;JUMP OUT IF FWI IS SET
                   (02426) *  TABLE FINISHED - OUTPUT DATA
A2F 064A2 20372037 (02427) STAB$OT: CLEAR(WI)                 ; LET APS INPUT PRG PROCEED
A30 064A4 02E03AA0 (02428)          R(A7)\ALIGN(A5)           ;QUANT VALUE\FIX CODE VALUE
A31 064A6 009C009C (02429)          MOV(R,OQ)\MOV(R,OQ)       ;OUTPUT QUANT AND CODED VALUES
A32 064A8 A000A000 (02430)          RETURN                    ;RETURN FROM CQ$SB
                   (02431) *
        00000033   (02432)          STAB$SZ=0A-STAB$SA        ;COMPUTE SIZE
                   (02433) *
    064AA          (02434)          END
```

```
                       (02435) * APS3 - STABS - 3-TAP STABILITY CHECK
                       (02436) *  APS PROGRAM FOR STAB - CHECKS STABILITY OF THE 3-TAP PITCH PREDICTOR
                       (02437) *              ALSO QUANTIZES AND CODES TAPS
                       (02438) *
                       (02439) *  INPUT AND OUTPUT STREAMS
                       (02440) *  IQ:  V(0),V(1),V(2),SA,2**-15
                       (02441) *       (CTHC*(N),DVLC*(N),N=1,...,LENGTH [N])
                       (02442) *
                       (02443) *  OQ:  (FLT PT QUANTIZED VALUE(I),FXD PT CODE(I),I=1,2,3)
                       (02444) *
                       (02445) *  HEADER BLOCK
                       (02446)          EVEN                         ;ENSURES HDR BLK IS DONE WITH FULL WORDS
    064AA 0000658C     (02447)          ADDR STABS$I                 ;POINTER TO CONSTRUCTION BLOCK
    064AC 000064C0     (02448)          ADDR STABS$+2*STABS$S        ;POINTER TO SCALAR BLOCK
    064AE 0001         (02449)          DATA 1                       ;SET NUMBER OF SCALARS
    064AF 0066         (02450)          DATA STABS$Z                 ;SET MODULE SIZE
    064B0 00006504     (02451)          ADDR STABS$A                 ;POINTER TO CHAIN ANCHOR
                       (02452)          EVEN                         ; ENSURE APS PRG DONE WITH FULL WORDS
                       (02453) *
                       (02454) *  END OF HEADER BLOCK
                       (02455) *
                       (02456) *  START OF APS PROGRAM
                       (02457) STABS$: BEGIN APS(STABS)
                       (02458) *
                       (02459) *  SET OUTPUT PROGRAM COUNTER , ALSO JUMP TO STAB$OP
                       (02460) *  AND START APU
A00 064B2 00202140     (02461)          JSN(STAB$OP,P2)
A01 064B4 02300030     (02462)          SET(RO)                      ;TURN ON OUTPUT ADDRESS GENERATION
                       (02463) * SET SUBROUTINE PROGRAM COUNTER (P1)
A02 064B6 04101740     (02464)          JSN(STAB$QC,P1)
                       (02465) *  LOAD BASE ADDRESS AND GENERATE V(I) ADDRESSES
A03 064B8 06C02000     (02466) 01:      LOAD(BR0,C2],TF)             ;LOAD V BASE ADDRESS & GEN V(0) ADDRESS
A04 064BA 08500000     (02467)          LOAD(BR1,M$$)                ;(UNUSED) SIZE MINUS 1
A05 064BC 0A8A0000     (02468)          ADD(BR0,M$$,TF)              ;GENERATE V(1) ADDRESS
A06 064BE 0C8AA006     (02469)          ADD(BR0,[10],TF)             ;GENERATE V(2) ADDRESS
                       (02470) *
                       (02471) *  INPUT SCALAR
A07 064C0 0EC20000     (02472) STABS$S: LOAD(BR0,M$$(1),TF)      ;GENERATE SA(1 TAP COEFF) ADDRESS
                       (02473) *
                       (02474) *  INPUT 2**-15
A08 064C2 10C203CE     (02475)          LOAD(BR0,SVT$UN1(1),TF)  ;GENERATE ADDRESS FOR 2**-15
                       (02476) *
                       (02477) *  INPUT CODING TABLES
A09 064C4 125297FE     (02478)          LOAD(BR1,CTHC1-W$(1))        ;BR1 POINTS TO THRESHOLD TABLE-1
A0A 064C6 146299FE     (02479)          LOAD(BR2,DVLC1-W$(1))        ;BR2 POINTS TO QUANT VALUE TABLE-1
A0B 064C8 16700006     (02480)          LOAD(BR3,8-2)                ;C1 - TABLE LENGTH = 8
A0C 064CA 18040020     (02481)          NOP(C)                       ;"CALL" STAB$QC FOR C1
A0D 064CC 1A52980E     (02482)          LOAD(BR1,CTHC2-W$(1))        ;BR1 POINTS TO THRESHOLD TABLE-1
A0E 064CE 1C629A0E     (02483)          LOAD(BR2,DVLC2-W$(1))        ;BR2 POINTS TO QUANT VALUE TABLE-1
A0F 064D0 1E70000E     (02484)          LOAD(BR3,16-2)               ;C2 - TABLE LENGTH = 16
A10 064D2 20040020     (02485)          NOP(C)                       ;"CALL STAB$QC FOR C2
A11 064D4 225297FE     (02486)          LOAD(BR1,CTHC3-W$(1))        ;BR1 POINTS TO THRESHOLD TABLE-1
A12 064D6 246299FE     (02487)          LOAD(BR2,DVLC3-W$(1))        ;BR2 POINTS TO QUANT VALUE TABLE-1
```

```
 A13 064D8 26700006 (02488)          LOAD(BR3,8-2)            ;C3 - TABLE LENGTH = 8
 A14 064DA 28040020 (02489)          NOP(C)                   ;"CALL" STABSQC FOR C3
 A15 064DC 2A200031 (02490)          CLEAR(RI)                ;DONE INPUT
 A16 064DE 2C000020 (02491)          NOP(0)
                    (02492) *
                    (02493) *  SUBROUTINE TO GENERATE ADDRESSES
                    (02494) *  FOR THRESHOLDS AND QUANTIZED VALUES.
                    (02495) *
 A17 064F0 2E000020 (02496) STABSQC: NOP(0)                   ;FOR JSN AND JUMP
 A18 064F2 3099003A (02497) 04:      ADDL(BR1,W$,TF)          ;GENERATE THRESH(I)
 A19 064E4 32A9003A (02498)          ADDL(BR2,W$,TF)          ;GENERATE QUANT VALUE(I)
 A1A 064E6 343910B1 (02499)          SUBL(BR3,1),JUMPP(04)    ;COUNT ENTRY AND JUMP IF MORE TABLE
 A1B 064E8 3699003A (02500)          ADDL(BR1,W$,TF)          ;DO LAST PAIR,AND ENSURE SET WI
 A1C 064EA 38A9107A (02501)          ADDL(BR2,W$,TF),JUMP(0A+1)
 A1D 064EC 3A300037 (02502)          SET(WI)                  ;SIGNAL END OF TABLE FOR APU
 A1E 064EE 3C040020 (02503)          NOP(C)
 A1F 064F0 3E001760 (02504)          JUMP(STABSQC)            ;RETURN AND SET P1 FOR NEXT CALL
 A20 064F2 40000020 (02505)          NOP(0)
                    (02506) *
                    (02507) *  OUTPUT PROGRAM
                    (02508) *
                    (02509) *  BW0: Y(I) FLT PT QUANTIZED VALUE OF C
                    (02510) *  BW1: U(I) FIXED POINT CODED VALUE OF C
                    (02511) *  BW2: LOOP COUNTER, COUNTS 3 OUTPUT PAIRS
                    (02512) *
 A21 064F4 42300032 (02513) STABSOP: SET(RA)                  ;TURN ON APU
 A22 064F6 44400038 (02514) 05:      LOAD(BW0,[0])            ;OUTPUT ARRAY Y
 A23 064F8 46600000 (02515)          LOAD(BW2,M$$)            ;(UNUSED)
 A24 064FA 48020000 (02516)          SUB(BW0,M$$)             ;PREPARE TO INCR LATER
 A25 064FC 4A501006 (02517) 06:      LOAD(BW1,[1])            ;OUTPUT ARRAY U
 A26 064FE 4C600000 (02518)          LOAD(BW2,M$$)            ;UBS-1
 A27 06500 4F120000 (02519)          SUB(BW1,M$$)             ;PREPARE TO INCR LATER
 A28 06502 500A0006 (02520) 07:      ADD(BW0,[0],TF)          ;QUANTIZED VALUE
 A29 06504 531A9002 (02521)          ADD(BW1,[9],TE)          ;CODE
 A2A 06506 54212081 (02522)          SUBL(BW2,1),JUMPP(07)    ;DONE?
 A2B 06508 56200030 (02523)          CLEAR(RO)                ;YES
 A2C 0650A 58000020 (02524)          NOP(0)
                    (02525) *
                    (02526) *  AFTER LAST SUBSTITUTABLE ARGUMENT
                    (02527) *  ASSIGN THE VALUE OF THE CHAIN ANCHOR
          00006504  (02528) STABS$A=0C
                    (02529) *
     0650C          (02530)          END                      ;END OF MODULE
                    (02531) *
                    (02532) *  STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                    (02533) *
     0650C 00000000 (02534) STABS$I DATA 6F'0.0'
      ...
                    (02535) *
                    (02536) *  COMPUTE MODULE SIZE
                    (02537) *
          00000066  (02538) STABS$Z=0L-STABS$
```

```
                 " APU3-INVPF(Y,A,U,V)    3-TAP PITCH INVERSE FILTER
                 ;  J. WOLF, 6/30/80
                 ;
                 ;  BINDS TO APS3-INVPF
                 ;
                 ;
                 ;IMPLEMENTS A 3-TAP PITCH INVERSE FILTER
                 ;  Y(N) = U(N) + V(0)*U(N-M+1) + V(1)*U(N-M) + V(2)*U(N-M-1)
                 ;                    N=0,1,...,YBS-1
                 ;
                 ;  BUFFER Y = FILTER OUTPUT
                 ;  REAL SCALAR A = PITCH PERIOD IN SAMPLES
                 ;  BUFFER U = CURRENT FILTER INPUT AND PREVIOUS 216 SAMPLES OF INPUT,
                 ;        WHERE THE INDEX FOR U RANGES FROM -216 TO YBS-1.  THAT IS, THE
                 ;        FIRST ENTRY IN U IS "LOGICAL" U(-216).  BUFFER U SHOULD BE
                 ;        (AT LEAST) 216+YBS SAMPLES LONG.
                 ;     NOTE: BUFFER U MUST BE COMPACT 32-BIT FLT PT
                 ;  BUFFER V = FILTER COEFFICIENTS
                 ;
                 ;WE PERFORM 2 FILTER CALCULATIONS IN THE 2 ADMS:
                 ;LEFT:
                 ;  Y(N)   = ((V(0)*U(N-M+1) + U(N))   + V(1)*U(N-M)) + V(2)*U(N-M-1)
                 ;RIGHT:
                 ;  Y(N+1) = ((V(1)*U(N-M+1) + U(N+1)) + V(2)*U(N-M)) + V(0)*U(N-M+2)
                 ;
                 ;INPUT SEQUENCE:
                 ;  (2**-15), SA, MOD. APS INSTR., V(0), V(1), V(2),
                 ;  U(-M+1),  U(0), U(1),   U(-M),  U(-M-1),  U(-M+2),
                 ;  U(2-M+1), U(2), U(3),   U(2-M), U(2-M-1), U(2-M+2), ...,
                 ;  U(N-M+1), U(N), U(N+1), U(N-M), U(N-M-1), U(N-M+2), ..., [PI]
                 ;OUTPUT SEQUENCE:
                 ; FXD PT PITCH, MOD. APS INSTR., 2 DUMMIES, Y(1), Y(2), ..., Y(UBS) [EO]
                 ;
                 ;DURATION:
                 ;  18 APU CYCLES = ~1.8 USEC PER 2 OUTPUT POINTS
                 ;  8 MEM REFS = ~1.44 USEC PER 2 OUTPUT POINTS (BUS 3)
                 ;
                 ;APU REGISTER USAGE:
                 ;        LEFT            RIGHT
                 ; A0     SUM             SUM
                 ; A1     V*Y             V*Y
                 ; A2     U(N)            U(N+1)
                 ; M0     U(N-M+1)        U(N-M+1)
                 ; M1     U(N-M)          U(N-M)
                 ; M2     U(N-M-1)        U(N-M+2)
                 ; M4     V(0)            V(1)
                 ; M5     V(1)            V(2)
                 ; M6     V(2)            V(0)
                 ;
                          EVEN
06518 0000                DATA INVPU$SA          ;APU START ADDRESS
06519 0020                DATA INVPU$SZ          ;SIZE
                 ;
```

```
PAGE   59: [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
             APU3-INVPF(Y,A,U,V)    3-TAP PITCH INVERSE FILTER

                        (02592) INVPU$: BEGIN APU(INVPF)
  A00 0651A 08E80000 (02593) INVPU$SA: MOV(IQA,M0) \ NOP            ;(2**-15)
  A01 0651C 08EC0000 (02594)          MOV(IQA,M4) \ NOP             ;PITCH (FLT PT)
  A02 0651E 84000000 (02595)          MUL(M0,M4) \ NOP              ;FIX THE PITCH
  A03 06520 08800000 (02596)          MOV(P,A0) \ NOP
  A04 06522 3A000000 (02597)          ALIGN(A0) \ NOP
  A05 06524 089C0000 (02598)          MOV(R,OQ) \ NOP               ;OUTPUT FIXED POINT PITCH
                        (02599) ;HERE WE WAIT ON OQE, PROCEED APS, READ/WRITE THE INSTRUCTION, WAIT
                        (02600) ;  AND PROCEED THE APS AGAIN, AND THEN GET DOWN TO THE COMPUTATION.
  A06 06526 901A0006 (02601) 01:      JUMPC(01,OQE)
  A07 06528 00000000 (02602)          NOP                     ;STORER SAYS WE NEED JUST 1 NOP HERE
  A08 0652A 20372037 (02603)          CLEAR(WI)               ;NOW LET THE APS PROCEED
  A09 0652C 08FC0000 (02604)          MOV(IQA,OQ) \ NOP       ;WRITE THE WHOLE INSTR. INTO THE APS
  A0A 0652E 901A000A (02605) 02:      JUMPC(02,OQE)           ;WAIT AGAIN
  A0B 06530 00000000 (02606)          NOP
  A0C 06532 20372037 (02607)          CLEAR(WI)               ;AND LET APS PROCEED AGAIN
  A0D 06534 08EC08EE (02608)          MOV(IQA,M4) \ MOV(IQA,M6)     ;FETCH V(0), V(1), V(2)
  A0E 06536 08ED08EC (02609)          MOV(IQA,M5) \ MOV(IQA,M4)
  A0F 06538 08EE08ED (02610)          MOV(IQA,M6) \ MOV(IQA,M5)
                        (02611) ;MAIN LOOP
  A10 0653A 08E808E8 (02612) 03:      MOV(IQA,M0)                   ;INPUT U(N-M+1)
  A11 0653C 84110411 (02613)          MOV(A1),MUL(M0,M4)            ;STORE 3RD PROD, START 1ST
  A12 0653E 41104110 (02614)          MOV(A0),ADD(A0,A1)            ;STORE 2ND SUM, START 3RD
  A13 06540 08F20000 (02615)          MOV(IQA,A2) \ NOP             ;INPUT U(N) \ U(N+1)
  A14 06542 000008F2 (02616)          NOP \ MOV(IQA,A2)
  A15 06544 089C089C (02617)          MOV(R,OQ)                     ;OUTPUT Y(N-2) \ Y(N-1)
  A16 06546 08E908E9 (02618)          MOV(IQA,M1)                   ;INPUT U(N-M)
  A17 06548 84810481 (02619)          MOV(A1),MUL(M1,M5)            ;STORE 1ST PROD, START 2ND
  A18 0654A 41404140 (02620)          ADD(A2,A1)                    ;START 1ST SUM
  A19 0654C 08EA0000 (02621)          MOV(IQA,M2) \ NOP             ;INPUT U(N-M-1) \ U(N-M+2)
  A1A 0654E 000008EA (02622)          NOP \ MOV(IQA,M2)
  A1B 06550 85518551 (02623)          MOV(A1),MUL(M2,M6)            ;STORE 2ND PROD, START 3RD
  A1C 06552 41104110 (02624)          MOV(A0),ADD(A0,A1)            ;STORE 1ST SUM, START 2ND
  A1D 06554 901C0010 (02625)          JUMPC(03,EO)                  ;LOOP UNLESS OUTPUT COMPLETE
  A1E 06556 20322032 (02626)          CLEAR(RA)
  A1F 06558 10000000 (02627)          JUMP(0)
            00000020 (02628) INVPU$SZ=0A-INVPU$SA
      0655A          (02629)          END
```

```
                      (02630) ' APS3-INVPF(V,A,U,V)        APS PROGRAM FOR INVPF
                      (02631) ;
                      (02632) ;  J. WOLF  7/1/80
                      (02633) ;
                      (02634) ;THIS APS PROGRAM MODIFIES ONE INSTRUCTION IN ITSELF BY:
                      (02635) ;  0. READING PITCH FROM SCALAR A, AND FIXING IT
                      (02636) ;  1. WRITING IT INTO THE RH OF THE BUS-1 COPY OF THE INSTRUCTION
                      (02637) ;  2. WAITING ON OQE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORER
                      (02638) ;     SAYS IS NECESSARY)
                      (02639) ;  3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
                      (02640) ;  4. WRITING IT TO APSMEM IN PSEUDOMEMORY
                      (02641) ;  5. WAITING FOR OQE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
                      (02642) ;     PROCEED.
                      (02643) ;
                      (02644) ;HEADER BLOCK
                      (02645) ;
                      (02646)         EVEN
    0655A 000065C2    (02647)         ADDR INVPS$I             ;PTR TO CONSTRUCTED INSTRUCTION BLOCK
    0655C 00006568    (02648)         ADDR INVPSI+2*INVPS$S    ;PTR TO SCALAR BLOCK
    0655E 0001        (02649)         DATA 1                   ;NUMBER OF SCALARS
    0655F 006E        (02650)         DATA INVPS$Z             ;MODULE SIZE
    06560 0000658A    (02651)         ADDR INVPS$A             ;PTR TO CHAIN ANCHOR
                      (02652)         EVEN
                      (02653) ;
                      (02654) ;INPUT PROGRAM
                      (02655) ;  BR0: V(I), THEN U(N-M+I)
                      (02656) ;  BR1: U(N)
                      (02657) ;  BR2: LOOP COUNTER, STARTS AT VBS-1
                      (02658) ;  BR3: SCRATCH
                      (02659) ;
                      (02660) INVPSI: BEGIN APS(INVPS)
A00 06562 00202440    (02661)         JSN(INVPSO,P2)           ;SET AND START OUTPUT PROGRAM
A01 06564 02300030    (02662)         SET(RO)
A02 06566 04C203CE    (02663)         LOAD(BR0,SVT$UN1(1),TF)  ;GEN (2**-15) ADR
A03 06568 06C20000    (02664) INVPS$S: LOAD(BR0,M$$(1),TF)    ;GEN SA ADDRESS (PITCH)
A04 0656A 08300037    (02665)         SET(WI)                  ;WAIT FOR APU TO SIGNAL WRITE COMPLETE
A05 0656C 0A000020    (02666)         NOP(0)
A06 0656E 0CC26584    (02667)         LOAD(BR0,INVPS$M(1),TF)  ;LOAD MODIFIED INSTR THRU APU INTO APS MEMORY
A07 06570 0E300037    (02668)         SET(WI)                  ;WAIT AGAIN FOR WRITE TO COMPLETE
A08 06572 10000020    (02669)         NOP
A09 06574 12C02000    (02670)         LOAD(BR0,[2],TF)         ;VBA, GEN V(0)
A0A 06576 14500000    (02671)         LOAD(BR1,M$$)            ;(UNUSED) VLNGTH-1
A0B 06578 160A0000    (02672)         ADD(BR0,M$$,TF)          ;V INCR, GEN V(1)
A0C 0657A 180AA006    (02673)         ADD(BR0,[10],TF)         ;GEN V(2)
                      (02674) ;NOW LOAD INFO FOR ACCESSING U(N-M+I) AND U(N+J)
A0D 0657C 1A401002    (02675)         LOAD(BR0,[1])            ;UBA
A0E 0657E 1C600000    (02676)         LOAD(BR2,M$$)            ;(UNUSED) UBS-1
A0F 06580 1E600000    (02677)         LOAD(BR2,M$$)            ;(UNUSED) UINC
A10 06582 200A0100    (02678)         ADD(BR0,216*V$)          ;ADD OFFSET FOR 216 PAST VALUES
                      (02679) ;NEXT INSTRUCTION GETS THE PITCH JAMMED INTO IT!
          000065B4    (02680) INVPS$M=$L
A11 06584 22600000    (02681)         LOAD(BR2,M$$)            ;LOAD PITCH
A12 06586 2429002D    (02682)         ADDB(BR2,BR2)            ;2*PITCH FOR FULLWORD ADDRESSING
```

```
PAGE   61:  [BBND]<DCA16>BBN16N.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APS3-INVPF(Y,A,U,V)        APS PROGRAM FOR INVPF

  A13 06588 26090025 (02683)          SUBB(BR0,BR2)           ;POINT TO U(B-N)
  A14 0658A 2850100E (02684)          LOAD(BR1,[1])           ;UBA
  A15 0658C 2A700000 (02685)          LOAD(BR3,N$$)           ;(UNUSED) UBS-1
  A16 0658E 2C120000 (02686)          SUB(BR1,N$$)            ;U INCR
  A17 06590 2E1A0100 (02687)          ADD(BR1,216*W$)         ;OFFSET FOR 216 PAST VALUES
  A18 06592 30700000 (02688)          LOAD(BR3,[0])           ;(UNUSED) YBA
  A19 06594 32600000 (02689)          LOAD(BR2,N$$)           ;YBS-1 FOR LOOP COUNTER
  A1A 06596 34700000 (02690)          LOAD(BR3,N$$)           ;(UNUSED) YINC
                     (02691) ;MAIN LOOP
  A1B 06598 3609003A (02692) $4:      ADDL(BR0,W$,TF)         ;U(B-N+1) (REMEMBER BUFFER U IS COMPACT)
  A1C 0659A 3899003A (02693)          ADDL(BR1,W$,TF)         ;U(B)
  A1D 0659C 3A99003A (02694)          ADDL(BR1,W$,TF)         ;U(B+1)
  A1E 0659E 3C090032 (02695)          SUBL(BR0,W$,TF)         ;U(B-N)
  A1F 065A0 3E090032 (02696)          SUBL(BR0,W$,TF)         ;U(B-N-1)
  A20 065A2 4009003E (02697)          ADDL(BR0,3*W$,TF)       ;U(B-N+2)
  A21 065A4 422910B2 (02698)          SUBL(BR2,2),JUMPP($4)   ;WE'RE DOING 2 POINTS AT ONCE
  A22 065A6 44200031 (02699)          CLEAR(RI)
  A23 065A8 46000020 (02700)          NOP(0)
                     (02701) ;
                     (02702) ;
                     (02703) ;OUTPUT PROGRAM
                     (02704) ;  BW0: Y(N)
                     (02705) ;  BW1: LOOP COUNTER (STARTS AT YBS-1)
                     (02706) ;  BW3: SCRATCH
                     (02707) ;
  A24 065AA 48300032 (02708) INVPSO: SET(RA)                  ;START APU
                     (02709) ;GENERATE BUS1 ADR FOR THE INSTRUCTION TO BE MODIFIED (RIGHT HALF)
  A25 065AC 4B7265B5 (02710)          LOAD(BW3,INVPS$N+N$(1),TE)
                     (02711) ;GENERATE PSEUDOMEMORY ADR FOR LOADING THE APS MEMORY
  A26 065AE 4CF3FFC0 (02712)          LOAD(BW3,APSMEM(1),TF)
                     (02713) ;NOW GET ON WITH THE "ORDINARY" OUTPUT
  A27 065B0 4EF20794 (02714)          LOAD(BW3,DMY$(1),TF)    ;OUTPUT 2 DUMMIES
  A28 065B2 50F20794 (02715)          LOAD(BW3,DMY$(1),TF)
  A29 065B4 52400022 (02716)          LOAD(BW0,[0])           ;YBA
  A2A 065B6 54500000 (02717)          LOAD(BW1,N$$)           ;YBS-1 FOR LOOP COUNTER
  A2B 065B8 56020000 (02718)          SUB(BW0,N$$)            ;Y INCR
                     (02719) ;OUTPUT LOOP
  A2C 065BA 580A0006 (02720) $5:      ADD(BW0,[8],TF)
  A2D 065BC 5A112CB1 (02721)          SUBL(BW1,1),JUMPP($5)
  A2E 065BE 5C200030 (02722)          CLEAR(RO)
  A2F 065C0 5E000020 (02723)          NOP(0)
      065C2          (02724)          END
            000065BA (02725) INVPS$A=$C                       ;ASSIGN VALUE TO CHAIN ANCHOR
      065C2 00000000 (02726) INVPS$I: DATA 7F*0.0'            ;BLOCK FOR CONSTRUCTED INSTRUCTIONS
       ...
            0000006E (02727) INVPS$Z=$L-INVPSI                ;COMPUTE BUS1 MODULE SIZE
```

```
                      * APU3-ADPEAK(U)   MONITOR PEAK ABS VAL OF INPUT SPEECH
                      ;  J WOLF, 10/30/80
                      ;
                      ;  BINDS TO APS3-ADPEAK
                      ;
                      ;PERFORMS A RUNNING MAXIMUM-ABSOLUTE-VALUE SCAN OF A BUFFER OF
                      ;  SPEECH DATA INPUT FROM THE ADAM, TO PERMIT MONITORING OF THE
                      ;  (PEAK) LEVEL OF THE INPUT SIGNAL.  THE RUNNING PEAK IS MAINTAINED
                      ;  AS AN INTEGER (VALUE = 0-2047), FOR EASE OF DISPLAY TO THE USER.
                      ;  THE A/D SIGNAL VALUES RANGE FROM -1 TO +1, SO THERE MUST BE BOTH
                      ;  A MAGNITUDE CHANGE (OF 2**11) AND AN INTEGER/FLTPT CHANGE
                      ;  WHEN THIS RUNNING MAX IS BROUGHT IN AND WRITTEN OUT AGAIN.
                      ;
                      ;INPUT SEQUENCE:
                      ;  PREVIOUS MAX (INTEGER), U(0), U(1), ..., U(UBS-1), [FI]
                      ;OUTPUT SEQUENCE:
                      ;  NEW MAX (INTEGER), [EO]
                      ;DURATION:
                      ;  2 ELEMENTS PER 7 APU CYCLES = ~.35 USEC/ELEMENT
                      ;  1 MEM REF PER ELEMENT = ~.5 USEC (BUS 1) PER ELEMENT
                      ;APU REGISTER USAGE:
                      ;           LEFT              RIGHT
                      ; A0:       U(N)(ODD)         U(N)(EVEN)
                      ; A1:       |MAX|(ODD)        |MAX|(EVEN)
                      ;DURING LOOP CLEANUP:
                      ; A2:       |MAX|(EVEN)
                      ; A3:       |MAX|(OVERALL)
                      ; A4:       |MAX|/16
                      ;
                                  EVEN
    065D0 0000                    DATA ADPKU$SA             ;APU START ADDRESS
    065D1 0012                    DATA ADPKU$SZ             ;APU MODULE SIZE
                      ;
                      ADPKU:  BEGIN APU(ADPKU)
A00 065D2 00F100F1    ADPKU$SA: MOV(IQA,A1)                 ;FETCH PREVIOUS MAX (INTEGER)
A01 065D4 22202220                INCRE(A1)                 ;MULT BY (2**15)*(2**-11)
A02 065D6 32313231                MOV(A1),NORM(A1)          ;AND NORMALIZE FOR GOOD MEASURE
A03 065D8 00910091                MOV(R,A1)                 ;A1 = PREVIOUS MAX (FLT PT)
A04 065DA 000000F0    B1:         NOP \ MOV(IQA,A0)         ;-- \ U(EVEN)
A05 065DC 00917100                MOV(R,A1) \ MAXABS(A0,A1)  ;STORE MAX(ODD) \ MAKE NEW MAX(EVEN)
A06 065DE 911D000A                JUMPS(EXMID,FI)           ;JUMP IF ODD NO. OF ELEMENTS IN U
A07 065E0 00F00000                MOV(IQA,A0) \ NOP         ;FETCH U(ODD) \ --
A08 065E2 71000091                MAXABS(A0,A1) \ MOV(R,A1)  ;MAKE NEW MAX(ODD) \ STORE MAX(EVEN)
A09 065E4 901D0004                JUMPC(B1,FI)              ;LOOP IF MORE DATA IN U
A0A 065E6 00910090    EXMID:      MOV(R,A1) \ MOV(R,EXO)    ;STORE MAX(ODD) \ BRING OVER MAX(EVEN)
A0B 065E8 00520000                MOV(EXI,A2) \ NOP         ;STORE MAX(EVEN)
A0C 065EA 62200000                MAX(A1,A2) \ NOP          ;RESOLVE MAX(U(ODD),U(EVEN))
A0D 065EC 2A730000                MOV(A3),DECRE(A3) \ NOP   ;MULT NEW MAX BY (2**-15)*(2**11)
A0E 065EE 3A940000                MOV(A4),ALIGN(A4) \ NOP   ;AND CONVERT TO INTEGER
A0F 065F0 009C0000                MOV(R,OQ) \ NOP           ;AND OUTPUT IT
A10 065F2 20322032                CLEAR(RA)                 ;AND HALT
A11 065F4 10000000                JUMP(0)
          00000012    ADPKU$SZ=0A-ADPKU$SA
```

```
PAGE   63:  [BBNB]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APU3-ADPEAK(U)   MONITOR PEAK ABS VAL OF INPUT SPEECH

      065F6          (02781)          END
```

```
                    (02782) ' APS3-ADPEAK
                    (02783) ;  J. WOLF, 10/30/80
                    (02784) ;  BINDS TO APU3-ADPEAK
                    (02785) ;
                    (02786) ;WE MAINTAIN THE RUNNING ABS. MAX. AS AN INTEGER IN THE INTEGER
                    (02787) ;  SCALAR TABLE, FOR CONVENIENCE IN DISPLAYING ITS VALUE TO THE USER.
                    (02788) ;  BECAUSE SNAP STANDARD BINDING DOESN'T ALLOW FOR INTEGER SCALARS
                    (02789) ;  AS ARGUMENTS TO ARRAY FUNCTIONS, WE SIMPLY "HARD BIND" THE
                    (02790) ;  INTEGER SCALAR OFFSET OF TADPK TO THIS APS PROGRAM.
                    (02791) ;
           00000057 (02792) TADPK=87
                    (02793) ;
                    (02794) ;HEADER BLOCK
                    (02795) ;
                    (02796)         EVEN
     065F6 0000661A (02797)         ADDR ADPKA$I            ;PTR TO CONSTRUCTED INSTRUCTION BLOCK
     065F8 00000000 (02798)         ADDR 0                  ;NO (REAL) SCALARS USED
     065FA 0000     (02799)         DATA 0
     065FB 0020     (02800)         DATA ADPKA$Z            ;APS MODULE SIZE
     065FC 0000660A (02801)         ADDR ADPKA$A            ;PTR TO CHAIN ANCHOR
                    (02802)         EVEN
                    (02803) ;INPUT PROGRAM:
                    (02804) ;  BR0: U(N)
                    (02805) ;  BR1: LOOP COUNTER
                    (02806) ;
                    (02807) ADPKA:
                    (02808) ADPKAI: BEGIN APS(ADPKA)
A00 065FE 00200440 (02809)          JSN(ADPKAD,P2)          ;SET AND START OUTPUT PROGRAM
A01 06600 02300030 (02810)          SET(RO)
A02 06602 05420559 (02811)          LOAD(BR0,ISVT$+TADPK(1),TE)  ;GEN ADR FOR TADPK
A03 06604 06401000 (02812) 010:     LOAD(BR0,E13)           ;LOAD U BASE ADR
A04 06606 08500000 (02813)          LOAD(BR1,N$$)           ;LOAD UBS-1
A05 06608 0A020000 (02814)          SUB(BR0,N$$)            ;UBASE-SPACING
A06 0660A 0C0A9006 (02815) 01:      ADD(BR0,E93,TF)         ;GEN U(N)
A07 0660C 0E190601 (02816)          SUBL(BR1,1),JUMPP(01)   ;LOOP TIL UBS ELEMENTS ARE DONE
A08 0660E 10200031 (02817)          CLEAR(RI)
A09 06610 12000020 (02818)          NOP(0)
                    (02819) ;
                    (02820) ;OUTPUT PROGRAM: WRITE NEW MAX BACK IN THE INTEGER SCALAR TABLE
                    (02821) ;
A0A 06612 14300032 (02822) ADPKAD: SET(RA)
A0B 06614 17420559 (02823)          LOAD(BW0,ISVT$+TADPK(1),TE)
A0C 06616 18200030 (02824)          CLEAR(RO)
A0D 06618 1A000020 (02825)          NOP(0)
                    (02826) ;
           0000660A (02827) ADPKA$A=$C                      ;CHAIN ANCHOR
     0661A          (02828)         END
     0661A 00000000 (02829) ADPKA$I: DATA 2F'0.0'           ;STORAGE FOR CONSTRUCTED INSTRUCTIONS
      ...
           00000020 (02830) ADPKA$Z = $L - ADPKA
```

```
                    (02831) ' APUDMP(V) -- DEBUGGING FUNCTION THAT DUMPS THE APU REGISTERS
                    (02832) ;  J. J. WOLF, 17 JULY 80
                    (02833) ;
                    (02834) ;BUFFER USAGE:
                    (02835) ;  V: COMPACT BUFFER, LENGTH 44 FULL WORDS, RECEIVES REGISTER CONTENTS
                    (02836) ;
                    (02837) ;APU PROGRAM: EMPTIES ALL REGISTERS INTO THE OQ
                    (02838) ;
                    (02839)         EVEN
    0661E 0000      (02840)         DATA DUMPU$SA
    0661F 001C      (02841)         DATA DUMPU$SZ
                    (02842) ;
                    (02843) DUMPU:  BEGIN APU(DUMPU)
                    (02844) DUMPU$SA:
                    (02845) ;ALL OUTPUTS ARE LEFT, THEN RIGHT
                    (02846) ;OUTPUT R, A0-A7
A00 06620 021C021C (02847)         MOV(OQ),R(A0)
A01 06622 023C023C (02848)         MOV(OQ),R(A1)
A02 06624 025C025C (02849)         MOV(OQ),R(A2)
A03 06626 027C027C (02850)         MOV(OQ),R(A3)
A04 06628 029C029C (02851)         MOV(OQ),R(A4)
A05 0662A 02BC02BC (02852)         MOV(OQ),R(A5)
A06 0662C 02DC02DC (02853)         MOV(OQ),R(A6)
A07 0662E 02FC02FC (02854)         MOV(OQ),R(A7)
A08 06630 009C009C (02855)         MOV(R,OQ)
                    (02856) ;OUTPUT P, THEN DO M0*M7, M1*M7, M2*M7, M3*M7 SO WE CAN (TRY TO)
                    (02857) ;  RECOVER THE CONTENTS OF M7, WHICH WE NEED TO CLOBBER
A09 06632 047C047C (02858)         MOV(OQ),MUL(M0,M7)
A0A 06634 04FC04FC (02859)         MOV(OQ),MUL(M1,M7)
A0B 06636 057C057C (02860)         MOV(OQ),MUL(M2,M7)
A0C 06638 05FC05FC (02861)         MOV(OQ),MUL(M3,M7)
A0D 0663A 00BC00BC (02862)         MOV(P,OQ)
                    (02863) ;NOW PUT 1.0 INTO M7 AND MULTIPLY TO OUTPUT M0-M3
A0E 0663C 16001600 (02864)         K(+1)
A0F 0663E 000F000F (02865)         MOV(R,M7)
A10 06640 04600460 (02866)         MUL(M0,M7)
A11 06642 04FC04FC (02867)         MOV(OQ),MUL(M1,M7)
A12 06644 057C057C (02868)         MOV(OQ),MUL(M2,M7)
A13 06646 05FC05FC (02869)         MOV(OQ),MUL(M3,M7)
                    (02870) ;PUT 1.0 INTO M0 AND MULTIPLY TO OUTPUT M4-M6
A14 06648 00080008 (02871)         MOV(R,M0)
A15 0664A 041C041C (02872)         MOV(OQ),MUL(M0,M4)
A16 0664C 043C043C (02873)         MOV(OQ),MUL(M0,M5)
A17 0664E 045C045C (02874)         MOV(OQ),MUL(M0,M6)
A18 06650 00BC00BC (02875)         MOV(P,OQ)
                    (02876) ;FINALLY, OUTPUT EXI
A19 06652 005C005C (02877)         MOV(EXI,OQ)
A1A 06654 20322032 (02878)         CLEAR(R4)             ;THAT'S ALL THERE IS TO SEE!
A1B 06656 10000000 (02879)         JUMP(0)
        0000001C (02880) DUMPU$SZ=$A-DUMPU$SA
    06658           (02881)         END
                    (02882) ;
                    (02883) ;APS MODULE FOR APUDMP
```

```
                    (02884) ;
                    (02885) ;NO INPUT STREAM!
                    (02886) ; OQ: V(0),V(1),...V(43)
                    (02887) ;
                    (02888)          EVEN
      06658 0000667A (02889)          ADDR DUMPS$I     ;PTR TO CONST. INSTR. BLOCK
      0665A 00000000 (02890)          ADDR 0           ;PTR TO SCALAR BLOCK
      0665C 0000     (02891)          DATA 0           ;NUMBER OF SCALARS
      0665D 001E     (02892)          DATA DUMPS$Z     ;MODULE SIZE IN HALFWORDS
      0665E 00006672 (02893)          ADDR DUMPS$A     ;PTR TO CHAIN ANCHOR
                    (02894)          EVEN
                    (02895) ;
                    (02896) ;APS INPUT PROGRAM
                    (02897) ;
                    (02898) DUMPS:  BEGIN APS(DUMPS)
                    (02899) DUMPSI:
 A00 06660 00200440 (02900)          JSN(P2,DUMPSO)          ;SET AND START OUTPUT PGM
 A01 06662 02300030 (02901)          SET(RO)
 A02 06664 04200031 (02902)          CLEAR(RI)               ;WE DO NO INPUT!
 A03 06666 06000020 (02903)          NOP(0)
                    (02904) ;
                    (02905) ;APS OUTPUT PROGRAM
                    (02906) ;
 A04 06668 00300032 (02907) DUMPSO: SET(RA)                 ;START APU
 A05 0666A 0A400000 (02908)          LOAD(BW0,[0])           ;YBA = OUTPUT BUFFER
 A06 0666C 0C500000 (02909)          LOAD(BW1,M$$)           ;(UNUSED)
 A07 0666E 0E020000 (02910)          SUB(BW0,M$$)            ;PREPARE TO INCR
 A08 06670 1050002B (02911)          LOAD(BW1,44-1)          ;THERE WILL BE 44 OUTPUTS
 A09 06672 128A0000 (02912) 01:     ADD(BW0,[0],TP)         ;OUTPUT LOOP
 A0A 06674 141109B1 (02913)          SUBL(BW1,1),JUMPP(01)
 A0B 06676 16200030 (02914)          CLEAR(RO)
 A0C 06678 18000020 (02915)          NOP(0)
     0667A          (02916)          END
          00006672 (02917) DUMPS$A=0C                  ;ASSIGN VALUE TO CHAIN ANCHOR
      0667A 00000000 (02918) DUMPS$I: DATA 2F"0.0"        ;BLOCK FOR CONSTRUCTED INSTRUCTIONS
       ...
          0000001E (02919) DUMPS$Z=0L-DUMPS            ;MODULE SIZE
```

```
                 * APU3 - HFC(Y,U,V)
                 * WHR 8/80
                 *
                 *  ROUTINE TO APPLY HFC(HIGH FREQUENCY CORRECTION)
                 *  TO THE AUTOCORRELATION COEFF.
                 *
                 *  INPUTS:
                 *       U(0),...,U(6)   = AUTOCORRELATION COEFFICIENTS
                 *       V(0),V(1),V(2)  = CONSTANT FACTORS
                 *                       = LAMDA*MU(I),I=0,1,2
                 *                         LAMDA=0.035
                 *                         MU(0)=0.375,MU(1)=-0.25,MU(2)=0.0625
                 *  OUTPUT:
                 *       Y(0),...,Y(6)   = HFC'D COEFFICIENTS
                 *                       = U(I) + LAMDA*MU(I)*E2,I=0,1,2
                 *               WHERE E2=MINIMUM MEAN-SQUARED PREDICTION ERROR
                 *                       = U(I),I=3,4,5,6
                 *  PROCEDURE
                 *
                 *  E2 IS COMPUTED FIRST IN THE FOLLOWING FOUR STEPS
                 *               COMPUTE K1=U(1)/U(0)
                 *               COMPUTE E1=(1-K1**2)*U(0)
                 *               COMPUTE K2=(U(2)-K1*U(1))/E1
                 *               COMPUTE E2=E1*(1-K2**2)
                 *  THEN THE Y(I)'S ARE COMPUTED AS ABOVE
                 *
                 *  APU REGISTER USAGE (ADM1 ONLY)
                 *
                 *  A0   SDIV(U0,E1),U(I)        M0      SDIV(1/U(0),1/E1
                 *  A1   SDIV(SCRATCH)           M1      1-K1**2
                 *  A2   U(2)                    M2      K1,1-K2**2
                 *  A3   K1**2                   M3      K2,E2
                 *  A4   K1*U(1)                 M4      SDIV(SCTATCH)
                 *  A5   K2**2,E2*V(I)           M5      U(1),K2
                 *  A6   SDIV(2)                 M6      SDIV(U(0),E1)
                 *  A7   1                       M7      K1,U(2)-K1*U(1),V(I)
                 *
                 *  INPUT AND OUTPUT SEQUENCE
                 *  IQ:
                 *       U(0),U(1),U(2),V(0),U(0),V(1),U(1),V(2),U(2),U(3),...,U(6)
                 *  OQ:
                 *       Y(0),...,Y(6)
                 *
                 *
                 *
                 *  HFC - APU INITIALIZATION
                 *
                         EVEN                    ;START ON FULL WORD BOUNDRY
0667E 0000               DATA    HFCU$SA         ;START ADDR AT SA
0667F 0040               DATA    HFCU$SZ         ;SIZE , COMPUTED AT END
                 *
                 HFCU$:  BEGIN APU(HFCU)         ;REFERENCE BEGIN STATMENT
                 *
```

```
                        (Ø2973) *  COMPUTE E2 - MINIMUM MEAN-SQUARED PREDICTION ERROR
                        (Ø2974) *  STEP #1 - COMPUTE K1
                        (Ø2975) *  INPUT U(Ø) - STORE IN M6 AND AØ FOR SDIV
AØØ Ø66ØØ ØØCEØØØØ (Ø2976) HFCU$A1: MOV(IQ,M6)\NOP                 ;M6=U(Ø)
AØ1 Ø6682 ØØFØØØØØ (Ø2977)         MOV(IQA,AØ)\NOP                 ;AØ=U(Ø)
                        (Ø2978) *  CALL SUBROUTINE TO DO SDIV OF U(Ø)
                        (Ø2979) *  1/U(Ø)  OUTPUT IN MØ
AØ2 Ø6684 3ØØØØØ28 (Ø298Ø)         CALL(HFCU$DV)                   ;COMPUTE 1/U(Ø)
                        (Ø2981) *  COMPUTE K1=U(1)/U(Ø)
AØ3 Ø6686 ØØEDØØØØ (Ø2982)         MOV(IQA,M5)\NOP                 ;M5=U(1)
AØ4 Ø6688 842ØØØØØ (Ø2983)         MUL(MØ,M5)\NOP                  ;P=U(1)/U(Ø)=K1
                        (Ø2984) *  STEP # 2 - COMPUTE E1
                        (Ø2985) *  COMPUTE 1-K1**2
AØ5 Ø668A 16ØØØØØØ (Ø2986)         K(1)\NOP                        ;R=1
AØ6 Ø668C ØØ97ØØØØ (Ø2987)         MOV(R,A7)\NOP                   ;A7=1.Ø
AØ7 Ø668E ØØAFØØØØ (Ø2988)         MOV(P,M7)\NOP                   ;M7=K1
AØ8 Ø669Ø ØØAAØØØØ (Ø2989)         MOV(P,M2)\NOP                   ;M2=K1
AØ9 Ø6692 856ØØØØØ (Ø299Ø)         MUL(M2,M7)\NOP                  ;P=K1*K1
AØA Ø6694 ØØB3ØØØØ (Ø2991)         MOV(P,A3)\NOP                   ;A3=K1**2
AØB Ø6696 4ECØØØØØ (Ø2992)         SUB(A7,A3)\NOP                  ;R=1-K1**2
AØC Ø6698 ØØ89ØØØØ (Ø2993)         MOV(R,M1)\NOP                   ;M1=1-K1**2
                        (Ø2994) *  COMPUTE E1 = (1-K1**2)*U(Ø)
AØD Ø669A B4CØØØØØ (Ø2995)         MUL(M1,M6)\NOP                  ;P=(1-K1**2)*U(Ø) = E1
                        (Ø2996) *  STEP #3 - COMPUTE K2
                        (Ø2997) *  COMPUTE 1/E1 - 1/E1 IN MØ
AØE Ø669C ØØBØØØØØ (Ø2998)         MOV(P,AØ)\NOP                   ;AØ=E1
AØF Ø669E ØØAEØØØØ (Ø2999)         MOV(P,M6)\NOP                   ;M6=E1
A1Ø Ø66AØ 3ØØØØØ28 (Ø3ØØØ)         CALL(HFCU$DV)                   ;COMPUTE 1/E1 - CALL SDIV
                        (Ø3ØØ1) *  COMPUTE U(2)-K1*U(1)
A11 Ø66A2 852ØØØØØ (Ø3ØØ2)         MUL(M2,M5)\NOP                  ;P=K1*U(1)
A12 Ø66A4 ØØF2ØØØØ (Ø3ØØ3)         MOV(IQA,A2)\NOP                        ;A2=U(2)
A13 Ø66A6 ØØB4ØØØØ (Ø3ØØ4)         MOV(P,A4)\NOP                   ;A4=K1*U(1)
A14 Ø66A8 4C4ØØØØØ (Ø3ØØ5)         SUB(A2,A4)\NOP                  ;R=U(2)-K1*U(1)
A15 Ø66AA ØØ8FØØØØ (Ø3ØØ6)         MOV(R,M7)\NOP                   ;M7=U(2)-K1*U(1)
                        (Ø3ØØ7) *  COMPUTE K2=(U(2)-K1*U(1))/E1
A16 Ø66AC 846ØØØØØ (Ø3ØØ8)         MUL(MØ,M7)\NOP                  ;P=K2=(U(2)-K1*U(1))/E1
                        (Ø3ØØ9) *  STEP #4 - COMPUTE E2
                        (Ø3Ø1Ø) *  COMPUTE K2**2
A17 Ø66AE ØØABØØØØ (Ø3Ø11)         MOV(P,M3)\NOP                   ;M3=K2
A18 Ø66BØ ØØADØØØØ (Ø3Ø12)         MOV(P,M5)\NOP                   ;M5=K2
A19 Ø66B2 85AØØØØØ (Ø3Ø13)         MUL(M3,M5)\NOP                  ;P=K2**2
A1A Ø66B4 ØØB5ØØØØ (Ø3Ø14)         MOV(P,A5)\NOP                   ;A5=K2**2
                        (Ø3Ø15) *  COMPUTE 1-K2**
A1B Ø66B6 4DEØØØØØ (Ø3Ø16)         SUB(A7,A5)\NOP                  ;R=1-K2**2
A1C Ø66B8 ØØ8AØØØØ (Ø3Ø17)         MOV(R,M2)\NOP                   ;M2=1-K2**2
                        (Ø3Ø18) *  COMPUTE E2
A1D Ø66BA 854ØØØØØ (Ø3Ø19)         MUL(M2,M6)\NOP                  ;P=E1*(1-K2**2)
A1E Ø66BC ØØABØØØØ (Ø3Ø2Ø)         MOV(P,M3)\NOP                   ;M3=E2
                        (Ø3Ø21) *  END OF E2 COMPUTATION
                        (Ø3Ø22) *
                        (Ø3Ø23) *  COMPUTE HIGH FREQUENCY CORRECTED
                        (Ø3Ø24) *  AUTOCORRELATION COEFF'S AND OUTPUT
                        (Ø3Ø25) *
```

```
A1F 0066BE 300F0039 (03026)          CALL(HFCU$CR)                    ;COMPUTE Y(0)
A20 0066C0 300A0039 (03027)          CALL(HFCU$CR)                    ;COMPUTE Y(1)
A21 0066C2 300A0039 (03028)          CALL(HFCU$CR)                    ;COMPUTE Y(2)
A22 0066C4 00FC0000 (03029)          MOV(IQA,OQ)\NOP                  ;OUTPUT Y(3)=U(3)
A23 0066C6 00FC0000 (03030)          MOV(IQA,OQ)\NOP                  ;OUTPUT Y(4)=U(4)
A24 0066C8 00FC0000 (03031)          MOV(IQA,OQ)\NOP                  ;OUTPUT Y(5)=U(5)
A25 0066CA 00FC0000 (03032)          MOV(IQA,OQ)\NOP                  ;OUTPUT Y(6)=U(6)
                    (03033) *
A26 0066CC 20322032 (03034)          CLEAR(RA)                        ;HALT APU
A27 0066CE 10000000 (03035)          JUMP(0)
                    (03036) *
                    (03037) *  SDIV SUBROUTINE - COMPUTES 1/C
                    (03038) *  C IS ASSUMED TO BE IN M6 AND A0
                    (03039) *
A28 0066D0 16A00000 (03040) HFCU$DV:  K(2)\NOP                  ;R=2.0
A29 0066D2 00960000 (03041)          MOV(R,A6)\NOP              ;A6=2.0
A2A 0066D4 2E000000 (03042)          RCP(A0)\NOP                ;R=1/C+DEL(=F0)
A2B 0066D6 00080000 (03043)          MOV(R,M0)\NOP              ;M0=F0
A2C 0066D8 04400000 (03044)          MUL(M0,M6)\NOP             ;P=F0*C
A2D 0066DA 00810000 (03045)          MOV(P,A1)\NOP              ;A1=F0*C
A2E 0066DC 49C00000 (03046)          SUB(A6,A1)\NOP             ;R=2-F0*C
A2F 0066DE 000C0000 (03047)          MOV(R,M4)\NOP              ;M4=R
A30 0066E0 04000000 (03048)          MUL(M0,M4)\NOP             ;P=F0*R(=F1)
A31 0066E2 00A00000 (03049)          MOV(P,M0)\NOP              ;M0=F1
A32 0066E4 04400000 (03050)          MUL(M0,M6)\NOP             ;P=F1*C
A33 0066E6 00810000 (03051)          MOV(P,A1)\NOP              ;A1=P
A34 0066E8 49C00000 (03052)          SUB(A6,A1)\NOP             ;R=2-F1*C
A35 0066EA 008C0000 (03053)          MOV(R,M4)\NOP              ;M4=R
A36 0066EC 04000000 (03054)          MUL(M0,M4)\NOP             ;P=F1*(2-F1*C)(=F2)
A37 0066EE 00A00000 (03055)          MOV(P,M0)\NOP              ;M0=F2=1/C
A38 0066F0 A000A000 (03056)          RETURN
                    (03057) *
                    (03058) *  SUBROUTINE FOR COMPUTING Y(I),I=0,1,2
                    (03059) *
A39 0066F2 00EF0000 (03060) HFCU$CR:  MOV(IQA,M7)\NOP                 ;INPUT V
A3A 0066F4 05E00000 (03061)          MUL(M3,M7)\NOP                   ;P=E2*V
A3B 0066F6 00F00000 (03062)          MOV(IQA,A0)\NOP                  ;INPUT U
A3C 0066F8 00B50000 (03063)          MOV(P,A5)\NOP                    ;A5=E2*V
A3D 0066FA 45000000 (03064)          ADD(A0,A5)\NOP                   ;R=Y=U+E2*V
A3E 0066FC 009C0000 (03065)          MOV(R,OQ)\NOP                    ;OUTPUT Y
A3F 0066FE A000A000 (03066)          RETURN
                    (03067) *
           00000040 (03068)          HFCU$SZ=$A-HFCU$SA               ;COMPUTE SIZE
                    (03069) *
    006700          (03070)          END
```

```
                        (03071) * APS3 - HFC(V,U,V)
                        (03072) * WHR 8/80
                        (03073) *
                        (03074) *  APS ROUTINE TO APPLY HIGH FREQUENCY CORRECTION
                        (03075) *  TO THE AUTOCORRELATION COEFFICIENTS.
                        (03076) *
                        (03077) *  INPUT STREAM:
                        (03078) *       U(0),U(1),U(2),V(0),U(0),V(1),U(1),V(2),U(2),U(3),...,U(6)
                        (03079) *  OUTPUT STREAM:
                        (03080) *       V(0),...,V(6)
                        (03081) *
                        (03082) *  HEADER BLOCK
                        (03083) *
                        (03084)         EVEN
      06700 00006746    (03085)         ADDR    HFCS$I              ;CONSTRUCT INSRT BLK
      06702 00000000    (03086)         ADDR    0                   ;SCALER BLK (NONE)
      06704 0000        (03087)         DATA    0                   ;# OF SCALERS=0
      06705 005A        (03088)         DATA    HFCS$Z              ;MODULE SIZE
      06706 00006740    (03089)         ADDR    HFCS$A              ;PTR TO CHAIN ANCHOR
                        (03090)         EVEN
                        (03091) *
                        (03092) *  END OF HEADER BLOCK
                        (03093) *
                        (03094) *  INPUT PROGRAM
                        (03095) *  REGISTER USAGE
                        (03096) *       BR0     U ADDR
                        (03097) *       BR1     U SIZE (UNUSED),INDEX
                        (03098) *       BR2     V ADDR
                        (03099) *       BR3     V SIZE (UNUSED)
                        (03100) *
                        (03101) *  START OF APS INPUT PROGRAM
                        (03102) *
                        (03103) HFCS$:  BEGIN   APS(HFCS)           ;START OF APS MODULE
                        (03104) *
A00 06708 00201540      (03105)         JSN(HFCS$OP,P2)             ;SET OUTPUT PC
A01 0670A 02300030      (03106)         SET(RO)                     ;START OUTPUT PROGRAM
                        (03107) *
                        (03108) *  LOAD AND OUTPUT U(0),U(1),U(2) ADDR'S
A02 0670C 04C01000      (03109) HFCS$1:  LOAD(BR0,[1],TF)           ;LOAD AND GEN U(0) ADDR
A03 0670E 06500000      (03110)         LOAD(BR1,M$$)               ;LOAD UBS-1(UNUSED)
A04 06710 088A0000      (03111)         ADD(BR0,M$$,TF)             ;GEN U(1) ADDR
A05 06712 0A8A9006      (03112)         ADD(BR0,[9],TF)             ;GEN U(2) ADDR
                        (03113) *
                        (03114) *  LOAD AND OUTPUT V ADDR'S
A06 06714 0CE02002      (03115) HFCS$2:  LOAD(BR2,[2],TF)           ;LOAD AND GEN V(0) ADDR
A07 06716 0E700000      (03116)         LOAD(BR3,M$$)               ;LOAD VBS-1 (UNUSED)
A08 06718 10220000      (03117)         SUB(BR2,M$$)                ;SUB SPACING , V(-1)
                        (03118) *
                        (03119) *  SET UP AND OUTPUT U(0) ADDR
A09 0671A 12029006      (03120)         SUB(BR0,[9])                ;MOV BACK TO U(1)
A0A 0671C 14829002      (03121)         SUB(BR0,[9],TF)             ;MOV BACK TO U(0) AND OUTPUT
                        (03122) *  SET UP AND OUTPUT V(1) ADDR
A0B 0671E 162AA002      (03123)         ADD(BR2,[10])               ;V(0) ADDR
```

```
PAGE   71:  [BBND]<DCA16>BBN16N.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APS3 - HFC(Y,U,V)

  A0C 06720 18AAA002 (03124)          ADD(BR2,[10],TF)              ;GEN V(1) ADDR
                     (03125) *  OUTPUT U(1),V(2) SEQUENCE
  A0D 06722 1A8A9002 (03126)          ADD(BR0,[9],TF)               ;GEN U(1) ADDR
  A0E 06724 1CAAA002 (03127)          ADD(BR2,[10],TF)              ;GEN V(2) ADDR
                     (03128) *  GEN REST OF U(I) ADDRS
  A0F 06726 1E500003 (03129)          LOAD(BR1,3)                   ;SET UP COUNTER
  A10 06728 208A9004 (03130) HFCS$3:  ADD(BR0,[9],TF)               ;GEN U(I) ADDR
  A11 0672A 221910B1 (03131)          SUBL(BR1,1),JUMPP(HFCS$3)     ;DEC INDEX AND JUMP
  A12 0672C 248A9004 (03132) HFCS$4:  ADD(BR0,[9],TF)               ;GEN U(6) ADDR
                     (03133) *
                     (03134) *  END OF APS INPUT PROGRAM
  A13 0672E 26200031 (03135)          CLEAR(RI)                     ;HALT INPUT
  A14 06730 28000020 (03136)          NOP(0)
                     (03137) *
                     (03138) *
                     (03139) *  OUTPUT APS PROGRAM
                     (03140) *  REGISTER USAGE
                     (03141) *        BW0     Y ADDR
                     (03142) *        BW1     Y SIZE (UNUSED)
                     (03143) *
  A15 06732 2A300032 (03144) HFCS$OP:  SET(RA)                      ;TURN ON APU
                     (03145) *
                     (03146) *  LOAD AND GEN Y ADDRS
  A16 06734 2CC00000 (03147) HFCS$5:  LOAD(BW0,[8],TF)              ;LOAD AND GEN Y(0)
  A17 06736 2E500000 (03148)          LOAD(BW1,N$$)                 ;YBS-1 (UNUSED)
  A18 06738 308A0000 (03149)          ADD(BW0,N$$,TF)               ;GEN Y(1) ADDR
                     (03150) *  GEN REST OF Y ADDRS
  A19 0673A 32500003 (03151)          LOAD(BW1,3)                   ;SET UP COUNTER
  A1A 0673C 348A8000 (03152) HFCS$6:  ADD(BW0,[8],TF)               ;GEN U(I) ADDR
  A1B 0673E 36111AB1 (03153)          SUBL(BW1,1),JUMPP(HFCS$6)             ;DEC INDEX AND JUMP
  A1C 06740 388A8004 (03154) HFCS$7:  ADD(BW0,[8],TF)               ;GEN Y(6) ADDR
                     (03155) *
                     (03156) *  END OF APS OUTPUT PROGRAM
  A1D 06742 3A200030 (03157)          CLEAR(RO)                     ;HALT OUTPUT
  A1E 06744 3C000020 (03158)          NOP(0)
                     (03159) *
            00006740 (03160) HFCS$A=0C                              ;CHAIN ANCHOR
                     (03161) *
      06746          (03162)          END
                     (03163) *
      06746 00000000 (03164) HFCS$I  DATA     14F'0.0'
       ...
            0000005A (03165) HFCS$Z=0L-HFCS$                        ;MODULE SIZE
```

```
                * APU3 - GAIN(Y,A,U,B,V,C)
                * WHR 9/80
                *
                *  ROUTINE TO COMPUTE QUANTIZER SEGMENT
                *  SCALE FACTORS. DELTA GAINS AND OVERALL GAIN
                *  ARE ALSO COMPUTED AND QUANTIZED FOR TX.
                *
                *  INPUTS:
                *       V(1),...,V(216)    = SECOND RESIDUAL SAMPLES
                *                        B = CONSTANT= 1/72
                *                        C = CONSTANT = 1/3
                *  OUTPUTS:
                *       U(0),U(1),U(2)     = CODED SEGMENT DELTA GAINS
                *       Y(0),Y(1),..,Y(5) = ALTERNATING SCALE FACTORS
                *                        AND 1/SCALE FACTORS
                *                        A = CODED OVERALL GAIN , G
                *
                *  PROCEDURE
                *       COMPUTE SUM(J) , J=1,2,3
                *               SUM(J)=SUM(V(I+N*72)**2),I=1,...72,N=0,1,2
                *       COMPUTE SG(J)=1/72*SUM(J) , J=1,2,3
                *       COMPUTE G = OVERALL GAIN
                *                 = 1/3*(SG(1)+SG(2)+SG(3))
                *       QUANTIZE G => GH - OUTPUT CODED G =>SA
                *       COMPUTE GH*GH
                *       COMPUTE 1/(GH*GH)
                *       COMPUTE DG(I)=SG(I)/(GH*GH)
                *       QUANTIZE DG(I) => DGH(I) - OUTPUT CODED DG => U(0,1,2)
                *       COMPUTE SCALE FACTOR Y(I)=GH*DGH(I) => Y(0,2,4)
                *       COMPUTE 1/SCALE FACTOR => Y(1,3,5)
                *
                *  INPUT AND OUTPUT SEQUENCE
                *       IQ: SB,SC,V(1),...,V(72),[FWI],V(73),...,V(144),[FWI],
                *               V(145),...,V(216),[FWI],QUANT AND CODING TABLES
                *               FOR G , DG(1) , DG(2) , DG(3)
                *       OQ: SA,U(0),Y(0),Y(1),U(1),Y(2),Y(3),U(2),Y(4),Y(5)
                *
                *  APU REGISTER USAGE
                *
                *  A0  SUM**2,HG**2,SDIV,Y(I)          M0  SUM**2,SUM(SG(I)),GH,SDIV
                *  A1  SG(1)                           M1  SUM(1),SG(1),DGH(1)
                *  A2  SG(2)                           M2  SUM(2),SG(2),DGH(2)
                *  A3  SG(3)                           M3  SUM**2,SUM(3),SG(3),DGH(3)
                *  A4  SG(1)+SG(2),SDIV                M4  SUM**2,SDIV
                *  A5  QUANT                           M5  1/72,GH
                *  A6  QUANT,SDIV                      M6  SUM**2,GH**2,SDIV,Y(I)
                *  A7  SUM**2,QUANT                    M7  1/3,1/(GH*GH)
                *
                *
                *  GAIN - APU INITIALIZATION
                *
                        EVEN                           ;START ON FULL WORD BOUNDRY
06762 0000              DATA    GAIN$SA                ;START ADDR AT SA
```

```
                  APU3 - GAIN(V,A,U,B,V,C)

     06763 0004     (03219)         DATA    GAIN$SZ                  ;SIZE , COMPUTED AT END
                    (03220) *
                    (03221) GAIN$:  BEGIN APU(GAIN)                  ;REF BEGIN STATMENT
                    (03222) *  INPUT CONSTANTS 1/72 AND 1/3
 A00 06764 00ED0000 (03223) GAIN$SA:  MOV(IQA,M5) \ NOP              ;M5=1/72
 A01 06766 00EF0000 (03224)         MOV(IQA,M7) \ NOP                ;M7=1/3
                    (03225) * COMPUTE SUM OF SQUARED SAMPLES
                    (03226) *  SEGMENT #1
 A02 06768 30000061 (03227)         CALL(GAIN$S)                     ;SUM (SAMPLES**2)
 A03 0676A 20372037 (03228)         CLEAR(WI)                        ;CONTINUE APS
 A04 0676C 00890000 (03229)         MOV(R,M1) \ NOP                  ;M1=SUM - SEGMENT #1
                    (03230) *  SEGMENT #2
 A05 0676E 30000061 (03231)         CALL(GAIN$S)                     ;SUM (SAMPLES**2)
 A06 06770 20372037 (03232)         CLEAR(WI)                        ;CONTINUE APS
 A07 06772 008A0000 (03233)         MOV(R,M2) \ NOP                  ;M2=SUM - SEGGENT #2
                    (03234) *  SEGMENT #3
 A08 06774 30000061 (03235)         CALL(GAIN$S)                     ;SUM (SAMPLES**2)
 A09 06776 20372037 (03236)         CLEAR(WI)                        ;CONTINUE APS
 A0A 06778 008B0000 (03237)         MOV(R,M3) \ NOP                  ;M3=SUM - SEGMENT #3
                    (03238) *  COMPUTE SG'S - SG(I)=1/72*SUM(I)
                    (03239) *  SEGMENT #1
 A0B 0677A 84A00000 (03240)         MUL(M1,M5) \ NOP                 ;P=SG(1)
 A0C 0677C 00B10000 (03241)         MOV(P,A1) \ NOP                  ;A1=SG(1)
 A0D 0677E 00A90000 (03242)         MOV(P,M1) \ NOP                  ;M1=SG(1)
                    (03243) *  SEGMENT #2
 A0E 06780 85200000 (03244)         MUL(M2,M5) \ NOP                 ;P=SG(2)
 A0F 06782 00B20000 (03245)         MOV(P,A2) \ NOP                  ;A2=SG(2)
 A10 06784 00AA0000 (03246)         MOV(P,M2) \ NOP                  ;M2=SG(2)
                    (03247) *  SEGMENT #3
 A11 06786 85A00000 (03248)         MUL(M3,M5) \ NOP                 ;P=SG(3)
 A12 06788 00B30000 (03249)         MOV(P,A3) \ NOP                  ;A3=SG(3)
 A13 0678A 00AB0000 (03250)         MOV(P,M3) \ NOP                  ;M3=SG(3)
                    (03251) *  COMPUTE OVERALL GAIN - G
 A14 0678C 42200000 (03252)         ADD(A1,A2) \ NOP                 ;R=SG(1)+SG(2)
 A15 0678E 00940000 (03253)         MOV(R,A4) \ NOP                  ;A4=SG(1)+SG(2)
 A16 06790 44600000 (03254)         ADD(A3,A4) \ NOP                 ;R=SG(1)+SG(2)+SG(3)
 A17 06792 00880000 (03255)         MOV(R,M0) \ NOP                  ;M0=R
 A18 06794 84600000 (03256)         MUL(M0,M7) \ NOP                 ;P=1/3(SG(1)+SG(2)+SG(3))
                    (03257) *  QUANTIZE OVERALL GAIN G = GH
 A19 06796 30000052 (03258)         CALL(GAIN$Q)                     ;QUANT G => GH
 A1A 06798 0000089C (03259)         NOP \ MOV(R,DQ)                  ;OUTPUT CODED G
                    (03260) * COMPUTE GH**2
 A1B 0679A 00880000 (03261)         MOV(R,M0) \ NOP                  ;M0=GH
 A1C 0679C 008D0000 (03262)         MOV(R,M5) \ NOP                  ;M5=GH
 A1D 0679E 84200000 (03263)         MUL(M0,M5) \ NOP                 ;P=GH*GH
                    (03264) *  COMPUTE 1/(GH*GH)
 A1E 067A0 00AE0000 (03265)         MOV(P,M6) \ NOP                  ;M6=GH*GH
 A1F 067A2 00B00000 (03266)         MOV(P,A0) \ NOP                  ;A0=GH*GH
 A20 067A4 30000042 (03267)         CALL(GAIN$DV)                    ;DIV - 1/(GH**2)
 A21 067A6 00AF0000 (03268)         MOV(P,M7) \ NOP                  ;M7=1/(GH*GH)
                    (03269) * SEGMENT #1
                    (03270) *  COMPUTE DG
 A22 067A8 84E00000 (03271)         MUL(M1,M7) \ NOP                 ;P=SG(1)/GH**2
```

```
                       (03272) *  QUANTIZE DG
 A23 067AA 30000052 (03273)         CALL(GAIN$Q)               ;QUANTIZE DG
 A24 067AC 0000089C (03274)         NOP \ MOV(R,DQ)            ;OUTPUT CODED DG
 A25 067AE 00090000 (03275)         MOV(R,M1) \ NOP            ;M1=DGH(1)
                       (03276) *  COMPUTE SCALE FACTOR V(1)
 A26 067B0 84A00000 (03277)         MUL(M1,M5) \ NOP           ;P=DGH(1)*GH
 A27 067B2 000C0000 (03278)         MOV(P,DQ) \ NOP            ;OUTPUT V(1)
                       (03279) *  COMPUTE 1/V(1)
 A28 067B4 00AF0000 (03280)         MOV(P,M6) \ NOP            ;M6=V(1)
 A29 067B6 00B00000 (03281)         MOV(P,A0) \ NOP            ;A0=V(1)
                       (03282) *
 A2A 067B8 30000042 (03283)         CALL(GAIN$DV)              ;SDIV - 1/V(1)
 A2B 067BA 000C0000 (03284)         MOV(P,DQ) \ NOP            ;OUTPUT 1/V(1)
                       (03285) *
                       (03286) * SEGMENT #2
                       (03287) *  COMPUTE DG
 A2C 067BC 85600000 (03288)         MUL(M2,M7) \ NOP           ;P=SG(2)/GH**2
                       (03289) *  QUANTIZE DG
 A2D 067BE 30000052 (03290)         CALL(GAIN$Q)               ;QUANTIZE DG
 A2E 067C0 0000089C (03291)         NOP \ MOV(R,DQ)            ;OUTPUT CODED DG
 A2F 067C2 000A0000 (03292)         MOV(R,M2) \ NOP            ;M2=DGH(2)
                       (03293) *  COMPUTE SCALE FACTOR V(2)
 A30 067C4 85200000 (03294)         MUL(M2,M5) \ NOP           ;P=DGH(2)*GH
 A31 067C6 000C0000 (03295)         MOV(P,DQ) \ NOP            ;OUTPUT V(2)
                       (03296) *  COMPUTE 1/V(2)
 A32 067C8 00AF0000 (03297)         MOV(P,M6) \ NOP            ;M6=V(2)
 A33 067CA 00B00000 (03298)         MOV(P,A0) \ NOP            ;A0=V(2)
                       (03299) *
 A34 067CC 30000042 (03300)         CALL(GAIN$DV)              ;SDIV - 1/V(2)
 A35 067CE 000C0000 (03301)         MOV(P,DQ) \ NOP            ;OUTPUT 1/V(2)
                       (03302) *
                       (03303) * SEGMENT #3
                       (03304) *  COMPUTE DG
 A36 067D0 85E00000 (03305)         MUL(M3,M7) \ NOP           ;P=SG(3)/GH**2
                       (03306) *  QUANTIZE DG
 A37 067D2 30000052 (03307)         CALL(GAIN$Q)               ;QUANTIZE DG
 A38 067D4 0000089C (03308)         NOP \ MOV(R,DQ)            ;OUTPUT CODED DG
 A39 067D6 000B0000 (03309)         MOV(R,M3) \ NOP            ;M3=DGH(3)
                       (03310) *  COMPUTE SCALE FACTOR V(3)
 A3A 067D8 85A00000 (03311)         MUL(M3,M5) \ NOP           ;P-DGH(3)*GH
 A3B 067DA 000C0000 (03312)         MOV(P,DQ) \ NOP            ;OUTPUT V(3)
                       (03313) *  COMPUTE 1/V(3)
 A3C 067DC 00AF0000 (03314)         MOV(P,M6) \ NOP            ;M6=V(3)
 A3D 067DE 00B00000 (03315)         MOV(P,A0) \ NOP            ;A0=V(3)
                       (03316) *
 A3E 067E0 30000042 (03317)         CALL(GAIN$DV)              ;SDIV - 1/V(3)
 A3F 067E2 000C0000 (03318)         MOV(P,DQ) \ NOP            ;OUTPUT 1/V(3)
                       (03319) *
 A40 067E4 20322032 (03320)         CLEAR(RA)                  ;HALT APU
 A41 067E6 10000000 (03321)         JUMP(0)
                       (03322)         EJECT
```

```
                  (03323) *  *  *  GAIN$DV - DIVIDE SUBROUTINE - COMPUTES 1/C
                  (03324) *  C IS ASSUMED TO BE IN M6 AND A0
                  (03325) *
A42 067E8 16A00000 (03326) GAIN$DV:  K(2)\NOP                ;R=2.0
A43 067EA 00960000 (03327)           MOV(R,A6)\NOP           ;A6=2.0
A44 067EC 2E000000 (03328)           RCP(A0)\NOP             ;R=1/C+DEL(=F0)
A45 067EE 00080000 (03329)           MOV(R,M0)\NOP           ;M0=F0
A46 067F0 84400000 (03330)           MUL(M0,M6)\NOP          ;P=F0*C
A47 067F2 00B40000 (03331)           MOV(P,A4)\NOP           ;A4=F0*C
A48 067F4 4CC00000 (03332)           SUB(A6,A4)\NOP          ;R=2-F0*C
A49 067F6 000C0000 (03333)           MOV(R,M4)\NOP           ;M4=R
A4A 067F8 84000000 (03334)           MUL(M0,M4)\NOP          ;P=F0*R(=F1)
A4B 067FA 00A00000 (03335)           MOV(P,M0)\NOP           ;M0=F1
A4C 067FC 84400000 (03336)           MUL(M0,M6)\NOP          ;P=F1*C
A4D 067FE 00B40000 (03337)           MOV(P,A4)\NOP           ;A4=P
A4E 06800 4CC00000 (03338)           SUB(A6,A4)\NOP          ;R=2-F1*C
A4F 06802 000C0000 (03339)           MOV(R,M4)\NOP           ;M4=R
A50 06804 84000000 (03340)           MUL(M0,M4)\NOP          ;P=F1*(2-F1*C)(=F2)
                  (03341) *  RESULTS =1/C IN P1
A51 06806 A000A000 (03342)           RETURN
                  (03343) *
                  (03344)            EJECT
```

```
PAGE   76:  [BBND]<DCA16>BBN16M.MSD.2, 29-Dec-80 14:30:40, Ed: WOLF
              APU3 - GAIN(Y,A,U,D,V,C)

                        (03345) *   *   *  GAINSQ - QUANTIZATION SUBROUTINE
                        (03346) *
  A52 0680C 000000F6 (03347) GAINSQ:  NOP \ MOV(IQA,A6)             ;INPUT 2**-15
                        (03348) *  PUT DATA IN A5 AND INIT CODE
  A53 0680A 00U50015 (03349)         MOV(P,A5)\MOV(ZERO,A5)   ;INPUT DATA \ INIT CODE=0
  A54 0680C 000002A0 (03350)         NOP\R(A5)                ;NOP\R=A5=ZERO
                        (03351) *  INNER LOOP - FIND QUANT VALUE AND CODE
  A55 0680E 00F60000 (03352) GAINSQ1:  MOV(IQA,A6)\NOP          ;INPUT BOUNDRY - TH(N)\NOP
  A56 06810 4EA046B5 (03353)         SUB(A5,A6)\MOV(A5),ADD(A5,A6)
                        (03354) *                                  ;(X-TH(N))\UPDATE CODE,CODE+1
  A57 06812 00F70000 (03355)         MOV(IQA,A7)\NOP          ;INPUT QUANTIZED VALUE L(N)\NOP
  A58 06814 00000000 (03356)         MOV(R,NULL)\NOP          ;ENSURE TB1 SET
  A59 06816 9116005F (03357)         JUMPS(GAINSQ3,FWI)       ;JUMP IF DONE TABLE
  A5A 06818 901E0055 (03358)         JUMPC(GAINSQ1,T1)        ;JUMP IF X > TH (KEEP LOOKING)
                        (03359) *  FOUND THE RIGHT QUANTIZATION BIN. READ IN REST OF TABLE.
  A5B 0681A 00E000E0 (03360) GAINSQ2:  MOV(IQA,NULL)            ;INPUT REST OF TABLE
  A5C 0681C 00000000 (03361)         NOP                      ;TO ALLOW FWI TO SET
  A5D 0681E 9016005B (03362)         JUMPC(GAINSQ2,FWI)       ;JUMP OUT IF FWI IS SET
                        (03363) *  TABLE FINISHED - OUTPUT DATA
  A5E 06820 20372037 (03364) GAINSQ3: CLEAR(WI)                 ; LET APS INPUT PRG PROCEED
  A5F 06822 02E03AA0 (03365)         R(A7)\ALIGN(A5)          ;QUANT VALUE\FIX CODE VALUE
                        (03366) *  QUANTIZED VALUE IN R1 - CODE IN R2
  A60 06824 A000A000 (03367)         RETURN                   ;RETURN FROM QUANTIZATION
                        (03368) *
                        (03369)         EJECT
```

```
                      (03370) *  *  *  GAIN$S - SUBROUTINE TO COMPUTE SUM OF SQUARES
                      (03371) *
                      (03372) *  INITIALIZE REGISTERS
A61 06026 00170017 (03373) GAIN$S:  MOV(ZERO,A7)                              ;A7=0,SUM
A62 06028 00000000 (03374)          MOV(ZERO,M3)                              ;MI=0
A63 0602A 05C005C0 (03375)          MUL(M3,M6)                                ;P=0
A64 0602C 02E002E0 (03376)          R(A7)                                     ;R=0
                      (03377) *
A65 0602E 00C005D0 (03378) GAIN$S1: MOV(IQ,M0) \ MOV(A0),MUL(M3,M6)          ;U0 IN ,P2=SQU3'
A66 06030 00EC4717 (03379)          MOV(IQA,M4) \ MOV(A7),ADD(A0,A7)          ;U0 IN ,R2=SUM1
A67 06032 00000000 (03380)          NOP
A68 06034 91160077 (03381)          JUMPS(GAIN$S2,FWI)                        ;EXIT IF U0 LAST
                      (03382) *
A69 06036 041000C0 (03383)          MOV(A0),MUL(M0,M4) \ MOV(IQ,M0)           ;P1=SQU0, U1 IN
A6A 06038 471700EC (03384)          MOV(A7),ADD(A0,A7) \ MOV(IQA,M4)          ;R1=SUM2,U1 IN
A6B 0603A 00000000 (03385)          NOP
A6C 0603C 91160079 (03386)          JUMPS(GAIN$S3,FWI)                        ;EXIT IF U1 LAST
                      (03387) *
A6D 0603E 00C00410 (03388)          MOV(IQ,M3) \ MOV(A0),MUL(M0,M4)           ;U2 IN ,P2=SQU1
A6E 06040 00EE4717 (03389)          MOV(IQA,M6) \ MOV(A7),ADD(A0,A7)          ;U2 IN, R2=SUM3
A6F 06042 00000000 (03390)          NOP
A70 06044 9116007B (03391)          JUMPS(GAIN$S4,FWI)                        ;EXIT IF U2 LAST
                      (03392) *
A71 06046 05D000C0 (03393)          MOV(A0),MUL(M3,M6) \ MOV(IQ,M3)           ;P1=SQU2, U3 IN
A72 06048 471700EE (03394)          MOV(A7),ADD(A0,A7) \ MOV(IQA,M6)          ;R1=SUM0, U3 IN
A73 0604A 00000000 (03395)          NOP
A74 0604C 90160065 (03396)          JUMPC(GAIN$S1,FWI)                        ;IF MORE INPUT CONT
                      (03397) *
                      (03398) *  CLEAN UP
                      (03399) *
A75 0604E 000005D0 (03400)          MOV(ZERO,M0) \ MOV(A0),MUL(M3,M6)         ;P2=SQU3
A76 06050 000C4717 (03401)          MOV(ZERO,M4) \ MOV(A7),ADD(A0,A7)         ;R2=SUM1
                      (03402) *
A77 06052 04100000 (03403) GAIN$S2: MOV(A0),MUL(M0,M4) \ MOV(ZERO,M0)        ;P1=SQU0
A78 06054 4717000C (03404)          MOV(A7),ADD(A0,A7) \ MOV(ZERO,M4)         ;R1=SUM2
                      (03405) *
A79 06056 00000410 (03406) GAIN$S3: MOV(ZERO,M3) \ MOV(A0),MUL(M0,M4)        ;P2=SQU1
A7A 06058 000E4717 (03407)          MOV(ZERO,M6) \ MOV(A7),ADD(A0,A7)         ;R2=SUM3
                      (03408) *
A7B 0605A 05D00000 (03409) GAIN$S4: MOV(A0),MUL(M3,M6) \ NOP                 ;P1=SQU2
A7C 0605C 47170000 (03410)          MOV(A7),ADD(A0,A7) \ NOP                  ;R1=SUM0
A7D 0605E 00000000 (03411)          MOV(P,A0)
A7E 06060 47174717 (03412)          MOV(A7),ADD(A0,A7)                        ;R1=SUM2, R2=SUM1
                      (03413) *
A7F 06062 00900090 (03414)          MOV(R,A0)                                 ;A01=SUM2,A02=SUM1
A80 06064 00900090 (03415)          MOV(R,EXO)                                ;EXI1=SUM1,EXI2=SUM2
A81 06066 00570057 (03416)          MOV(EXI,A7)                               ;A71=SUM1,A72=SUM2
A82 06068 47004700 (03417)          ADD(A0,A7)                                ;ADD BOTH SIDES
                      (03418) *  RESULTS (TOTAL SUM) IN R1 AND R2
A83 0606A A000A000 (03419)          RETURN
                      (03420) *
          00000004 (03421)          GAIN$SZ=.A-GAIN$SA                        ;COMPUTE SIZE
                      (03422) *
```

```
              APU3 - GAIN(V,A,U,B,V,C)

     0606C          (03423)         END
                    (03424) *
```

```
                         (03425) * APS3 - GAIN(Y,A,U,B,V,C)
                         (03426) * WHR 9/80
                         (03427) *
                         (03428) *  APS ROUTINE FOR GAIN
                         (03429) *
                         (03430) *  INPUT STREAM:
                         (03431) *  IQ:
                         (03432) *
                         (03433) *  OUTPUT STREAM:
                         (03434) *  OQ:
                         (03435) *
                         (03436) *  INPUT PROGRAM REGISTER USAGE
                         (03437) *       BR0     SA,V(BA),2**-15
                         (03438) *       BR1     SB,V SIZE(UNUSED),CTH*(BA)
                         (03439) *       BR2     SC,COUNTER 72,DVL*(BA)
                         (03440) *       BR3     COUNTER 3,COUNTER (TABLE LENGTH)
                         (03441) *
                         (03442) *  HEADER BLOCK
                         (03443)         EVEN                                 ;START HDR BLK ON FULL WORDS
    066C 000060F4        (03444)         ADDR GAINS$I               ;POINTER TO CONSTRUCTION BLOCK
    066E 0000607A        (03445)         ADDR GAINS$+2*GAINS$S      ;POINTER TO SCALER BLOCK
    0670 0003            (03446)         DATA 3                     ;SET NUMBER OF SCALERS
    0671 000E            (03447)         DATA GAINS$Z               ;SET MODULE SIZE
    0672 000060EC        (03448)         ADDR GAINS$A               ;POINTER TO CHAIN ANCHOR
                         (03449)         EVEN                                 ;START APS PRG ON FULL WORDS
                         (03450) *
                         (03451) *  END OF HEADER BLOCK
                         (03452) *
                         (03453) *  START OF APS PROGRAM
                         (03454) GAINS$: BEGIN APS(GAINS)
                         (03455) *  SET OUTPUT PROGRAM COUNTER , ALSO JUMP TO GAINS$OP
                         (03456) *  AND START APU
A00 06074 00203240       (03457)         JSN(GAINS$OP,P2)                     ;SET OUTPUT PC
A01 06076 02300030       (03458)         SET(RO)                              ;TURN ON OUTPUT ADDR GENERATION
A02 06078 04102040       (03459)         JSN(GAINS$Q,P1)                      ;SET P1 TO GAINS$Q
                         (03460) *
                         (03461) *  INPUT SCALERS
A03 0607A 06420000       (03462) GAINS$S:  LOAD(BR0,M$$(1))                   ;LOAD SA ADDR
A04 0607C 00D20000       (03463)         LOAD(BR1,M$$(1),TF)                  ;GEN SB ADDR(1/72)
A05 0607E 0AE20000       (03464)         LOAD(BR2,M$$(1),TF)                  ;GEN SC ADDR(1/3)
                         (03465) *  MOVE SA TO OUTPUT REGISTER
A06 06080 0C010011       (03466)         MOVB(BW0,BR0)                        ;BW0=SA
                         (03467) *  LOAD 2ND RESIDUAL SAMPLES BASE ADDR - V
A07 06082 0E402000       (03468)         LOAD(BR0,[2])                        ;LOAD V BA
A08 06084 10500000       (03469)         LOAD(BR1,M$$)                        ;UNUSED (SIZE - 1)
A09 06086 12020000       (03470)         SUB(BR0,M$$)                         ;PREPARE TO INCREMENT LATER
                         (03471) *  LOAD COUNTERS IN BR2 AND BR3
A0A 06088 14700002       (03472)         LOAD(BR3,3-1)                        ;BR3=# OF SEGMENTS - 1
A0B 0608A 16600046       (03473) GAINS$1:  LOAD(BR2,72-2)                     ;BR2=# OF SAMPLES/SEG - 2
                         (03474) *  LOOP FOR SEGMENT SAMPLES INPUT
A0C 0608C 180AA00A       (03475) GAINS$2:  ADD(BR0,[10],TF)                   ;GEN V(I) ADDR
A0D 0608E 1A290C01       (03476)         SUBL(BR2,1),JUMPP(GAINS$2)           ;DEC INDEX AND JUMP
A0E 06090 1C090F7A       (03477)         ADDL(BR0,M$,TF),JUMP(BA+1)           ;DO LAST VALUE AND ENSURE SET WI
```

```
A0F 06092 1E300037 (03478)        SET(WI)                        ;SIGNAL END OF INPUT
A10 06094 20000020 (03479)        NOP(0)
A11 06096 22390081 (03480)        SUBL(BR3,1),JUMPP(GAINS$1)     ;LOOP FOR 2ND AND 3RD SEGMENT
                   (03481) *
                   (03482) *
                   (03483) *  INPUT CODING TABLES - AND 2**-15
                   (03484) *    FOR OVERALL GAIN - G
A12 06098 24C203CE (03485)        LOAD(BR0,SVTSUN1(1),TF)        ;GEN ADDR FOR 2**-15
A13 0609A 2652996E (03486)        LOAD(BR1,CTHG-W$(1))     ;BR1 POINTS TO THRESHOLD TABLE-1
A14 0609C 2862986E (03487)        LOAD(BR2,DVLG-W$(1))     ;BR2 POINTS TO QUANT VALUE TABLE-1
A15 0609E 2A70003E (03488)        LOAD(BR3,64-2)           ;G - TABLE LENGTH = 64
A16 060A0 2C040020 (03489)        NOP(C)                   ;"CALL" GAINS$Q FOR G
                   (03490) *    FOR DG(1)
A17 060A2 2EC203CE (03491)        LOAD(BR0,SVTSUN1(1),TF)        ;GEN ADDR FOR 2**-15
A18 060A4 305299EE (03492)        LOAD(BR1,CTHDG-W$(1))    ;BR1 POINTS TO THRESHOLD TABLE-1
A19 060A6 32629BEE (03493)        LOAD(BR2,DVLDG-W$(1))    ;BR2 POINTS TO QUANT VALUE TABLE-1
A1A 060A8 34700002 (03494)        LOAD(BR3,4-2)            ;DG - TABLE LENGTH = 4
A1B 060AA 36040020 (03495)        NOP(C)                   ;"CALL GAINS$Q FOR DG
                   (03496) *    FOR DG(2)
A1C 060AC 38C203CE (03497)        LOAD(BR0,SVTSUN1(1),TF)        ;GEN ADDR FOR 2**-15
A1D 060AE 3A5299EE (03498)        LOAD(BR1,CTHDG-W$(1))    ;BR1 POINTS TO THRESHOLD TABLE-1
A1E 060B0 3C629BEE (03499)        LOAD(BR2,DVLDG-W$(1))    ;BR2 POINTS TO QUANT VALUE TABLE-1
A1F 060B2 3E700002 (03500)        LOAD(BR3,4-2)            ;DG - TABLE LENGTH = 4
A20 060B4 40040020 (03501)        NOP(C)                   ;"CALL" GAINS$Q FOR DG
                   (03502) *    FOR DG(3)
A21 060B6 42C203CE (03503)        LOAD(BR0,SVTSUN1(1),TF)        ;GEN ADDR FOR 2**-15
A22 060B8 445299EE (03504)        LOAD(BR1,CTHDG-W$(1))  ;BR1 POINTS TO THRESHOLD TABLE-1
A23 060BA 46629BEE (03505)        LOAD(BR2,DVLDG-W$(1))    ;BR2 POINTS TO QUANT VALUE TABLE-1
A24 060BC 48700002 (03506)        LOAD(BR3,4-2)            ;DG - TABLE LENGTH = 4
A25 060BE 4A040020 (03507)        NOP(C)                   ;"CALL" GAINS$Q FOR DG
A26 060C0 4C200031 (03508)        CLEAR(RI)                ;DONE INPUT
A27 060C2 4E000020 (03509)        NOP(0)
                   (03510) *
                   (03511) *  SUBROUTINE TO GENERATE ADDRESSES
                   (03512) *  FOR THRESHOLDS AND QUANTIZED VALUES.
                   (03513) *
A28 060C4 50000020 (03514) GAINS$Q: NOP(0)                 ;FOR JSN AND JUMP
A29 060C6 5299003A (03515) 04:    ADDL(BR1,W$,TF)          ;GENERATE THRESH(I)
A2A 060C8 54A9003A (03516)        ADDL(BR2,W$,TF)          ;GENERATE QUANT VALUE(I)
A2B 060CA 563929B1 (03517)        SUBL(BR3,1),JUMPP($4)    ;COUNT ENTRY AND JUMP IF MORE TABLE
A2C 060CC 5099003A (03518)        ADDL(BR1,W$,TF)          ;DO LAST PAIR,AND ENSURE SET WI
A2D 060CE 5AA92E7A (03519)        ADDL(BR2,W$,TF),JUMP($A+1)
A2E 060D0 5C300037 (03520)        SET(WI)                  ;SIGNAL END OF TABLE FOR APU
A2F 060D2 5F040020 (03521)        NOP(C)
A30 060D4 60002860 (03522)        JUMP(GAINS$Q)            ;RETURN AND SET P1 FOR NEXT CALL
A31 060D6 62000020 (03523)        NOP(0)
                   (03524) *
                   (03525)        EJECT
```

```
                APS3 - GAIN(V,A,U,B,Y,C)

                       (03526) *
                       (03527) *  OUTPUT APS PROGRAM
                       (03528) *
                       (03529) *  RESISTER USAGE
                       (03530) *        BW0      SA , V BASE ADDR
                       (03531) *        BW1      U BASE ADDR
                       (03532) *        BW3      U SIZE-1
                       (03533) *
A32 06080 64300032     (03534) GAINS$OP:  SET(RA)                    ;TURN ON APU
                       (03535) *
                       (03536) *  GENERATE SCALAR A ADDR
A33 060DA 60010010     (03537)       MOVB(BW0,BW0,TF)                ;GEN SA ADDR
                       (03538) *                                     ;(CODED OVERALL GAIN)
                       (03539) *  LOAD SCALE FACTOR & INVERSE ADDR
A34 060DC 68400058     (03540)       LOAD(BW0,[0])          ;V BASE ADDR
A35 060DE 6A700000     (03541)       LOAD(BW3,M$$)                   ;(UNUSED)
A36 060E0 6C020000     (03542)       SUB(BW0,M$$)                    ;PREPARE TO INCR LATER
                       (03543) *
                       (03544) *  LOAD DELTA GAIN CODE ADDR
A37 060E2 6E501006     (03545)       LOAD(BW1,[1])                   ;U BASE ADDR
A38 060E4 70700000     (03546)       LOAD(BW3,M$$)                   ;UBS-1
A39 060E6 72120000     (03547)       SUB(BW1,M$$)                    ;PREPARE TO INCR LATER
                       (03548) *  LOOP FOR OUTPUT OF U AND V
A3A 060E8 751A9006     (03549) 01:   ADD(BW1,[9],TE)                 ;OUTPUT ADDR OF CODED DG
A3B 060EA 760A8002     (03550)       ADD(BW0,[8],TF)                 ;OUTPUT ADDR OF SCALE FAC
A3C 060EC 780A8002     (03551)       ADD(BW0,[8],TF)                 ;OUTPUT ADDR OF 1/SCALE FAC
A3D 060EE 7A313A01     (03552)       SUBL(BW3,1),JUMPP(01)           ;DEC COUNT AND JUMP
A3E 060F0 7C200030     (03553)       CLEAR(RO)                       ;DONE
A3F 060F2 7E000020     (03554)       NOP(0)
                       (03555) *
          000068EC     (03556) GAINS$A=0C                            ;CHAIN ANCHOR
                       (03557) *
    060F4              (03558)       END
                       (03559) *
    060F4 00000000     (03560) GAINS$T DATA    7F'0.0'
     ...
                       (03561) *
          0000000E     (03562) GAINS$Z=0L-GAINS$                     ;MODULE SIZE
```

```
                 (03563) * SPECIAL SUPPORT MODULES FOR FEOF & FEOI
                 (03564) *       FORWARD & INVERSE EVEN-ODD SEPARATE MODULES.
                 (03565) *       ALSO CALLED BY HOST FOR FFTHR AND FFIHR.
                 (03566) *       (ALSO CALLED BY HOST FOR FFTLR AND FFILR.)
                 (03567) *                                   15 MAY 79
                 (03568) *                                   J F BURNS
                 (03569) *
                 (03570) *
                 (03571) * ENTER HERE FOR FEOI
                 (03572) *
06902 90600024   (03573) FEOI$SSN MOVIR  R6, FLG$SET + FLG$G0     SET-G0 LITERAL
06904 2003       (03574)          NOP    FEOF$1                 & SAVE IT
06905 0000       (03575)          EVEN
                 (03576) *
                 (03577) * ENTER HERE FOR FEOF
                 (03578) *
06906 90600004   (03579) FEOF$SSN MOVIR  R6, FLG$CLR + FLG$G0     CLR-G0 LITERAL
                 (03580) *                                      & SAVE IT
                 (03581) *
                 (03582) * COMMON PATH FOR FEOS SET-UP
                 (03583) *
06908 F0420004   (03584) FEOF$1   MOVMR  R4, 4 * H$ (R1)         GET W-ID
0690A 9A40FF00   (03585)          ANDIR  R4, MSK$LBYT
0690C 3C47       (03586)          LRS    R4, 7                  CNVRT TO BCT INDEX
0690D 0000       (03587)          EVEN
0690E F0580687   (03588)          MOVMR  R5, BCT$AT + H$ (R4)   GET PWR-OF-2 BS
06910 2651       (03589)          INCR   R5, 1                  CNVRT TO REAL BS
06911 3A52       (03590)          LLS    R5, 2                  CNVRT TO COS TBL INDEX
06912 90403190   (03591)          MOVIR  R4, COS$               ADDR OF APS CONST
06914 C64A23FA   (03592)          PUSHMIL R4, SINCOS$T (R5)     LOAD .5COS(PI/N)
06916 C64A23FC   (03593)          PUSHMIL R4, SINCOS$T + H$(R5) LOAD .5SIN(PI/N)
                 (03594) *
06918 B610DEFA   (03595)          CALL   R1, AP$BNDR            NORMAL BIND, BUT G0 = 0
                 (03596) *
0691A E066000C   (03597)          MOVRM  R6, AP$G0 (R3)         SET OR CLEAR G0 CNTRL
                 (03598) *
0691C 0E70       (03599)          RETURN
                 (03600) *
0691D 0000       (03601)          EVEN
```

```
                      * APU3 - FEUSL   FOURIER EVEN-ODD SEPARATE LONG (MAP-300)
                      *
                      *PUTS/TAKES FS/2 PT FROM ONE PT BEYOND END OF BUFFER
                      *       BINDS TO APS3-F11AL
                      *
                      *
                                     EVEN
     0691E 0000                      DATA      FEOSL$SA
     0691F 0064                      DATA      FEOSL$SZ
                      *
                      FEOSL$          BEGIN APU(FEOS)
           00000000                  $A=0
                      *
                      *  INITIALIZE PROGRAM REGISTERS
                      *
A00 06920 16A016A0    FEOSL$SA    K(2)                               R=2
A01 06922 00EA00EA                MOV(IQA,M2)                        M2=1/2COS(TH)
A02 06924 00EB00EB                MOV(IQA,M3)                        M3=1/2SIN(TH)
A03 06926 000F000F                MOV(R,M7)                          M7=2
A04 06928 05600560                MUL(M2,M7)                         P=2*M2
A05 0692A 05EA05EA                MOV(M2),MUL(M3,M7)                 M2=C(=COS(TH))
A06 0692C 00AB00AB                MOV(P,M3)                          M3=S(=SIN(TH))
A07 0692E 000D000D                MOV(ZERO,M5)                       M5=0(FIRST SIGN VALUE)
A08 06930 91040012                JUMPS(FEOSL$1,G0)                  JUMP TO INVERSE IF G0 SET
                      *
                      *FORWARD-FIRST SAMPLE PROCESS
                      *
A09 06932 16001600                K(1)                               R=1
A0A 06934 000C000C                MOV(R,M4)                          M4=1(FIRST COS FOR FWD)
A0B 06936 00F000F0                MOV(IQA,A0)                        A0=R(0)
A0C 06938 00F100F1                MOV(IQA,A1)                        A1=I(0)
A0D 0693A 410049B0                ADD(A0,A1) \ SUB(A0,A1)            R1=R(0)+I(0) (=YR(0))\
                      ;                                              R(0)-I(0)(=YI(0))
A0E 0693C 40104010                MOV(A0),ADD(A0,A0)                 DOUBLE FOR SCALE
A0F 0693E 009C009C                MOV(R,OQ)                          OUT=2YR0\2YR(N-1)        FOR FWD
A10 06940 001C001C              MOV(ZERO,OQ)             OUT=2YI0\2YI(N-1)
A11 06942 10J00019                JUMP(FEOSL$2)
                      *
                      *INVERSE - FIRST SAMPLE PROCESS
                      *
                      FEOSL$1
A12 06944 06000600                K(-1)                              R=-1
A13 06946 000C000C                MOV(R,M4)                          M4=-1(FIRST COS FOR INV)
A14 06948 00F000F0                MOV(IQA,A0)                        A0=R(0)
A15 0694A 00F100F1                MOV(IQA,A1)                        A1=I(0)
A16 0694C 410049B0                ADD(A0,A1) \ SUB(A0,A1)            R1=R(0)+I(0)\R0-I0
A17 0694E 009C009C                MOV(R,OQ)                          OUT=2YR0\2YI0 FOR INV
A18 06950 1000003A                JUMP(FEOSL$3)
                      *
                      *  MAIN PROGRAM LOOP FOR FORWARD  (K=1,2,....N/2)
                      *
                      FEOSL$2
A19 06952 05A005B0                MUL(M3,M5) \ MUL(M3,M4)            S*SIN(K-1)\S*COS(K-1)
```

```
PAGE   84:  [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
               APU3 - FEOSL   FOURIER EVEN-ODD SEPARATE LONG (MAP-300)

  A1A 06954 00F000F0 (03655)         MOV(IQA,A0)                   A0=R(K)
  A1B 06956 00F100F1 (03656)         MOV(IQA,A1)                   A1=R(N-K)
  A1C 06958 41004900 (03657)         ADD(A0,A1) \ SUB(A0,A1)       R=2UR\2VI
  A1D 0695A 00F200F2 (03658)         MOV(IQA,A2)                   A2=IK
  A1E 0695C 00F300F3 (03659)         MOV(IQA,A3)                   A3=I(N-K)
  A1F 0695E 85158535 (03660)         MOV(A5),MUL(M2,M4) \ MOV(A5),MUL(M2,M5)
                     (03661) ;                                     A5=S*SIN(K-1)\S*COS(K-1)
  A20 06960 00940089 (03662)         MOV(R,A4) \ MOV(R,M1)         A4=2UR\2VI
  A21 06962 43584B58 (03663)         MOV(EXO),ADD(A2,A3) \ MOV(EXO),SUB(A2,A3)
                     (03664) ;                                     EXO=2UR\2VI
  A22 06964 00490054 (03665)         MOV(EXI,M1) \ MOV(EXI,A4)
                     (03666) ;                                     M1=2VI\A4=2UR
  A23 06966 00800097 (03667)         MOV(R,M0) \ MOV(R,A7)         M0=2VR\A7=2UI
  A24 06968 00B60086 (03668)         MOV(P,A6)                     A6=C*COS(K-1)\C*SIN(K-1)
  A25 0696A 4DD845D8 (03669)         MOV(EXO),SUB(A6,A5) \ MOV(EXO),ADD(A6,A5)
                     (03670) *                                     EXO=2VR\2UI
  A26 0696C 00570048 (03671)         MOV(EXI,A7) \ MOV(EXI,M0)     A7=2UI\M0=2VR
  A27 0696E 008C008D (03672)         MOV(R,M4) \ MOV(R,M5)         M4=COS\M5=SIN
  A28 06970 840084A0 (03673)         MUL(M0,M4) \ MUL(M1,M5)       P=2VR*COS\2VI*SIN
  A29 06972 00980098 (03674)         MOV(R,EXO)                    EXO=COS\SIN
  A2A 06974 004D004C (03675)         MOV(EXI,M5) \ MOV(EXI,M4)
                     (03676) *                                     M5=SIN\M4=COS
  A2B 06976 00B800B8 (03677)         MOV(P,EXO)                    EXO=2VR*COS\2VI*SIN
  A2C 06978 84300491 (03678)         MOV(A0),MUL(M0,M5) \ MOV(A1),MUL(M1,M4)
                     (03679) *                                     A0=2VR*COS\A1=2VI*SIN
  A2D 0697A 00510050 (03680)         MOV(EXI,A1) \ MOV(EXI,A0)     A1=2VI*SIN\A0=2VR*COS
  A2E 0697C 49004900 (03681)         SUB(A0,A1)                    R=2VRCOS-2VISIN
  A2F 0697E 42924A92 (03682)         MOV(A2),ADD(A4,A2) \ MOV(A2),SUB(A4,A2)
                     (03683) *                                     A2=2VRCOS-2VISIN
  A30 06980 00B000B1 (03684)         MOV(P,A0) \ MOV(P,A1)         A0=2VR*SIN\A1=2VI*COS
  A31 06982 00B800B8 (03685)         MOV(P,EXO)                    EXO=2VRSIN\2VICOS
  A32 06984 00510050 (03686)         MOV(EXI,A1) \ MOV(EXI,A0)     A1=2VICOS\A0=2VRSIN
  A33 06986 411C411C (03687)         MOV(OQ),ADD(A0,A1) \ MOV(OQ),ADD(A0,A1)
                     (03688) *                                     OUT=2VR(K)\2VI(K)
  A34 06988 4AF242F2 (03689)         MOV(A2),SUB(A7,A2) \ MOV(A2),ADD(A7,A2)
                     (03690) *                                     A2=2VRSIN+2VICOS
  A35 0698A 02520A52 (03691)         MOV(A2),R(A2) \ MOV(A2),NEG(A2)
                     (03692) *                                     A2=2VI(K)\-2VI(N-K)
  A36 0698C 009C009C (03693)         MOV(R,OQ) \ MOV(R,OQ)         OUT=2VI(K)\2VI(N-K)
  A37 0698E 901C0019 (03694)         JUMPC(FEOSL$2,EO)
                     (03695) *
  A38 06990 00000000 (03696)         NOP
  A39 06992 10000059 (03697)         JUMP(FEOSL$4)
                     (03698) *
                     (03699) *  MAIN LOOP FOR INVERSE (K=1,2,...N/2)
                     (03700) *
                     (03701) FEOSL$3
  A3A 06994 85A085B0 (03702)         MUL(M3,M5) \ MUL(M3,M4)       S*SIN(K-1)\S*COS(K-1)
  A3B 06996 00F000F0 (03703)         MOV(IQA,A0)                   A0=R(K)
  A3C 06998 00F100F1 (03704)         MOV(IQA,A1)                   A1=R(N-K)
  A3D 0699A 41004900 (03705)         ADD(A0,A1) \ SUB(A0,A1)       R=2UR\2VI
  A3E 0699C 00F200F2 (03706)         MOV(IQA,A2)                   A2=IK
  A3F 0699E 00F300F3 (03707)         MOV(IQA,A3)                   A3=I(N-K)
```

```
A40 069A0 05150535 (03708)          MOV(A5),MUL(M2,M4) \ MOV(A5),MUL(M2,M5)
                   (03709) *                                   A5=S*SIN(K-1)\S*COS(K-1)
A41 069A2 00940089 (03710)          MOV(R,A4) \ MOV(R,M1)      A4=2UR\2VI
A42 069A4 43584058 (03711)          MOV(EXO),ADD(A2,A3) \ MOV(EXO),SUB(A2,A3)
                   (03712) *                                   EXO=2UR\2VI
A43 069A6 00490054 (03713)         MOV(EXI,M1) \ MOV(EXI,A4)
                   (03714) *                                   M1=2VI\A4=2UR
A44 069A8 00800097 (03715)          MOV(R,M0) \ MOV(R,A7)      M0=2VR\A7=2UI
A45 069AA 00B60086 (03716)          MOV(P,A6)                  A6=C*COS(K-1)\C*SIN(K-1)
A46 069AC 45D840D8 (03717)          MOV(EXO),ADD(A6,A5) \ MOV(EXO),SUB(A6,A5)
                   (03718) *                                   EXO=2VR\2UI
A47 069AE 00570040 (03719)          MOV(EXI,A7) \ MOV(EXI,M0)  A7=2UI\M0=2VR
A48 069B0 008C008D (03720)          MOV(R,M4) \ MOV(R,M5)      M4=COS\M5=SIN
A49 069B2 840084A0 (03721)          MUL(M0,M4) \ MUL(M1,M5)    P=2VR*COS\2VI*SIN
A4A 069B4 00980098 (03722)          MOV(R,EXO)                 EXO=COS\SIN
A4B 069B6 004D004C (03723)          MOV(EXI,M5) \ MOV(EXI,M4)
                   (03724) *                                   M5=SIN\M4=COS
A4C 069B8 00B800B8 (03725)          MOV(P,EXO)                 EXO=2VR*COS\2VI*SIN
A4D 069BA 84300491 (03726)          MOV(A0),MUL(M0,M5) \ MOV(A1),MUL(M1,M4)
                   (03727) *                                   A0=2VR*COS\A1=2VI*SIN
A4E 069BC 00510050 (03728)          MOV(EXI,A1) \ MOV(EXI,A0)  A1=2VI*SIN\A0=2VR*COS
A4F 069BE 49004900 (03729)          SUB(A0,A1)                 R=2VRCOS-2VISIN
A50 069C0 42924A92 (03730)          MOV(A2),ADD(A4,A2) \ MOV(A2),SUB(A4,A2)
                   (03731) }                                   A2=2VRCOS-2VISIN
A51 069C2 00B000B1 (03732)          MOV(P,A0) \ MOV(P,A1)      A0=2VR*SIN\A1=2VI*COS
A52 069C4 00B800B8 (03733)          MOV(P,EXO)                 EXO=2VRSIN\2VICOS
A53 069C6 00510050 (03734)          MOV(EXI,A1) \ MOV(EXI,A0)  A1=2VICOS\A0=2VRSIN
A54 069C8 411C411C (03735)          MOV(OQ),ADD(A0,A1) \ MOV(OQ),ADD(A0,A1)
                   (03736) *                                   OUT=2VR(K)\2VI(K)
A55 069CA 4AF242F2 (03737)          MOV(A2),SUB(A7,A2) \ MOV(A2),ADD(A7,A2)
                   (03738) *                                   A2=2VRSIN+2VICOS
A56 069CC 0252 0A52 (03739)         MOV(A2),R(A2) \ MOV(A2),NEG(A2)
                   (03740) *                                   A2=2VI(K)\-2VI(N-K)
A57 069CE 009C009C (03741)          MOV(R,OQ) \ MOV(R,OQ)      OUT=2VI(K)\2VI(N-K)
A58 069D0 901C003A (03742)          JUMPC(FEOSL$3,EO)
                   (03743) *
A59 069D2 20322032 (03744) FEOSL$4   CLEAR(RA)           HALT
A5A 069D4 00000000 (03745)          NOP
A5B 069D6 10000000 (03746)          JUMP(0)
A5C 069D8 00000000 (03747)          NOP
A5D 069DA 00000000 (03748)          NOP
A5E 069DC 00000000 (03749)          NOP
A5F 069DE 00000000 (03750)          NOP
A60 069E0 00000000 (03751)          NOP
A61 069E2 00000000 (03752)          NOP
A62 069E4 00000000 (03753)          NOP
A63 069E6 10000000 (03754)          JUMP(FEOSL$SA)
                   (03755) *
          00000064 (03756)          FEOSL$SZ=$A-FEOSL$SA
    069E8          (03757)          END FEOSL$SZ
                   (03758)          EVEN
                   (03759) *
```

```
                         (03760) * APS3 - F11AL   APS PROGRAM FOR FEOSL
                         (03761) *
                         (03762) *
                         (03763) *INPUT STREAM-SA, SB; R(0),I(0); R(K),R(N-K),I(K),I(N-K),...FI
                         (03764) *OUTPUT STREAM-VR(0),VI(0); VR(K),VR(N-K),VI(K),VI(N-K),...EO
                         (03765) *       WHERE K=1,2,...N/2
                         (03766) *     FOR INVERSE
                         (03767) *
                         (03768) *OUTPUT STREAM FOR FORWARD- VR(K),VR(N-1-K),VI(K),VI(N-1-K));...
                         (03769) *     FOR K=0,1,...(N-1)/2
                         (03770) *
                         (03771) *
                         (03772)             EVEN                     START ON WORD BOUNDARY
     069E8 00006A60      (03773)             ADDR F11AL$CI
     069EA 000069F0      (03774)             ADDR F11AL$
     069EC 0000          (03775)             DATA 0
     069ED 0000          (03776)             DATA F11AL$SZ
     069EE 00006A4C      (03777)             ADDR F11AL$CA
                         (03778)             EVEN
                         (03779) *
                         (03780) F11AL$      BEGIN APS(F11AL)
                         (03781) *
           00000002      (03782)          URI = 2
           00000002      (03783)          VRI = 2
                         (03784) *INPUT PROGRAM
                         (03785) *
A00 069F0 00201C40 (03786) 01          JSN(F11AL$3,P2)       SET OUTPUT PC
A01 069F2 02300030 (03787)             SET(RO)              TURN ON OUTPUT
A02 069F4 04C23190 (03788)             LOAD(BR0,COS$(1),TF)  GEN SA ADDRESS (CONTAINS 1/2 COS)
A03 069F6 06C23192 (03789)             LOAD(BR0,SIN$(1),TF)  GEN SB ADDRESS (CONTAINS 1/2 SIN)
A04 069F8 08C03000 (03790)             LOAD(BR0,[3],TF)          LOAD INPUT BASE(UBASE),R(0)
A05 069FA 0A600000 (03791)             LOAD(BR2,M$$)        LOAD VECTOR-U SIZE (MINUS 1)
A06 069FC 0C500000 (03792)             LOAD(BR1,M$$)          SET SPACING(UNUSED)
A07 069FE 0E0A8006 (03793)             ADD(BR0,[1])       ADDR OF R(1)
A08 06A00 10000DE4 (03794)             JUMPS(F1,G0)         JUMPIF INV
A09 06A02 12020002 (03795)             SUB(BR0,URI,TF)      GEN I(0) ADDR
A0A 06A04 14080002 (03796)             ADD(BR0,URI)         RESET PTR TO REAL(1)
A0B 06A06 16390011 (03797) F1        MOVB(BR3,BR0)          UBASE+4
A0C 06A08 18390020 (03798)             ADDB(BR3,BR2)        UBASE+(N-1)+4
A0D 06A0A 1A39002D (03799)             ADDB(BR3,BR2)        UBASE+2(N-1)+4
A0E 06A0C 1C39002D (03800)             ADDB(BR3,BR2)        UBASE+3(N-1)+4
A0F 06A0E 1E39002D (03801)             ADDB(BR3,BR2)        UBASE+4(N-1)+4
A10 06A10 20001244 (03802)             JUMPC(F2,G0)           JUMP IF FORWARD
A11 06A12 22090017 (03803)             MOVB(BR3,BR3,TF)          R(N) TREATED AS I(0)
                         (03804) *
                         (03805) *INPUT ADDRESS GENERATION LOOP  (K=1,2,...N/2)
                         (03806) *
A12 06A14 24090034 (03807) F2        SUBL(BR0,4)      MAKE IT POINT TO R(0)
A13 06A16 260A8018 (03808) 02          ADD(BR0,[1],TF)      GEN R(K)
A14 06A18 2802D002 (03809)             SUB(BR3,[1],TF)      GEN R(N-K)
A15 06A1A 2A0A0002 (03810)             ADD(BR0,URI,TF)      GEN I(K)
A16 06A1C 2CBA0002 (03811)             ADD(BR3,URI,TF)      GEN I(N-K)
A17 06A1E 2E020002 (03812)             SUB(BR0,URI)         RESET TO REAL OF K
```

```
A18 06A20 30320002 (03013)          SUB(BR3,URI)           RESET TO REAL OF N-K
A19 06A22 322913B2 (03014)          SUBL(BR2,2),JUMPP(02)  CONTINUE UNTIL LAST ELEMENT (N-2-2K=>0)
                   (03015) *
A1A 06A24 34200031 (03016)          CLEAR(RI)              HALT INPUT
                   (03017) *
A1B 06A26 36000020 (03018)        NOP(0)
                   (03019) *
                   (03020) *OUTPUT PROGRAM
                   (03021) *
                   (03022) *
A1C 06A28 38300032 (03023) F11AL$3    SET(RA)
A1D 06A2A 3A403012 (03024)          LOAD(BW0,C3])          LOAD VECTOR-Y BASE ADDRESS
A1E 06A2C 3C500000 (03025)          LOAD(BW1,N$$)          LOAD VECTOR-Y SIZE
A1F 06A2E 3E600000 (03026)          LOAD(BW2,N$$)           SET TO SPACING (UNUSED)
A20 06A30 4011003A (03027)        ADDL(BW1,2)               N+1
A21 06A32 42310010 (03028)          MOVB(BW3,BW0)        YBASE
A22 06A34 4431002A (03029)          ADDB(BW3,BW1)            YBASE+(N+)
A23 06A36 4631002A (03030)          ADDB(BW3,BW1)            YBASE+2(N+1)
A24 06A38 4831002A (03031)          ADDB(BW3,BW1)           YBASE+3(N+1)
A25 06A3A 4A31002A (03032)          ADDB(BW3,BW1)           YBASE+4(N+1)=YEND+4=YR(N+1)
A26 06A3C 4C010034 (03033)        SUBL(BW0,4)       R(-1)
A27 06A3E 4E002DA4 (03034)          JUMPC(F4,G0)             JUMP IF FORWARD
A28 06A40 500AB016 (03035)          ADD(BW0,C11],TF)          YR(0) IF INV
A29 06A42 520A0002 (03036)           ADD(BW0,YRI,TF)          YI(0) IF INV
A2A 06A44 54020002 (03037)          SUB(BW0,YRI)         POINT TO YR(0) IF INV
A2B 06A46 56310034 (03038)        SUBL(BW3,4)               YEND IF INV
A2C 06A48 58110032 (03039)          SUBL(BW1,2)            N-1
                   (03040) *
                   (03041) *
                   (03042) *OUTPUT ADDRESS GENERATION LOOP
                   (03043) *
                   (03044) *
A2D 06A4A 5A0AB00A (03045) F4       ADD(BW0,C11],TF)       GEN YR(K)
A2E 06A4C 5C02B002 (03046)          SUB(BW3,C11],TF)       GEN YR(N-K)
A2F 06A4E 5E0A0002 (03047)          ADD(BW0,YRI,TF)        GEN YI(K)
A30 06A50 600A0002 (03048)          ADD(BW3,YRI,TF)        GEN YI(N-K)
A31 06A52 62020002 (03049)          SUB(BW0,YRI)           RESET TO REAL OF K
A32 06A54 64370002 (03050)          SUB(BW3,YRI)           RESET TO REAL OF (N-K)
A33 06A56 66112DB2 (03051)          SUBL(BW1,2),JUMPP(F4)    CONTINUE UNTIL LAST ELEMENT
A34 06A58 68200030 (03052)          CLEAR(RO)              HALT OUTPUT
A35 06A5A 6A000020 (03053)        NOP(0)
A36 06A5C 6C000020 (03054)          NOP(0)
A37 06A5E 6E000020 (03055)          NOP(0)
                   (03056) *
                   (03057) *
          00006A4C (03058) F11AL$CA=0C  ASSIGN A VALUE TO CHAIN ANCHOR
                   (03059) *
    06A60          (03060)          END  0A-1          END OF MODULE
                   (03061) *
                   (03062) *
                   (03063) * STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                   (03064) *
                   (03065) *
```

```
06A60 00000000 (03866) FIIAL$CI    DATA     0F"0.0"
...
               (03867) *
               (03868) *
      00000000 (03869) FIIAL$SZ=   #L-FIIAL$        COMPUTE MODULE SIZE
               (03870) *
               (03871) *
               (03872)        EVEN
               (03873) *
```

```
" APU3 - APC(Y,A,U,V,W,R,S,T)   APC (WITH NOISE SHAPING)
; KFIELD 8/80
;
; RUNS WITH APB3-APC (APCS$)
;
; FUNCTION:
;       COMPUTE CODED APC RESIDUAL FROM PREEMPHASIZED
;       INPUT SPEECH.  (SEE DETAILED DOCUMENTATION OF
;       APC LOOP IN ALGORITHM SPECIFICATION.)
;
; PARAMETERS:
;       OUT:    Y(FIXED): CODED RESIDUAL SAMPLES
;       IN:     A(REAL):  PITCH LAG (SA)
;       IN:     U(REAL):  PREEMPHASIZED SPEECH
;       IN:     V(REAL):  FILTER COEFFS: 6 SPECTRAL, 3 PITCH, 6 N.S.
;       IN:     W(REAL):  3 QUANTIZER SEGMENT SCALE FACTORS,
;                         ALTERNATING WITH THEIR RECIPROCALS
;       IN/OUT: R(REAL):  PITCH FILTER HISTORY: LAST, CURRENT FRAMES
;       IN/OUT: S(REAL):  SPECTRAL FILTER HISTORY
;       IN/OUT: T(REAL):  NOISE SHAPING FILTER HISTORY
;
; SYMBOLS USED IN COMMENTS IN THIS MODULE:
; I:               INDEX OF CURRENT ITERATION OF APC LOOP (0:VBS-1)
; M:               PITCH LAG (IN SAMPLES) (SA)
; ANS(1:6)         NOISE SHAPING FILTER COEFFS (ZEROS) (V(9:14))
; AH(1:6)          SPECTRAL FILTER (ALSO N.S. FILTER POLES) COEFFS (V(0:5))
; C1H              PITCH FILTER COEFF #1 (V(6))
; C2H              PITCH FILTER COEFF #2 (V(7))
; C3H              PITCH FILTER COEFF #3 (V(8))
; Q1(I-6:I-1)      NOISE SHAPING FILTER INPUT (T(0:5))
; VH(I-6:I-1)      SPECTRAL FILTER INPUT (S(0:5))
; RH(I-M+1)        PITCH FILTER INPUT #1 (MOST RECENT) (RH(0)=R(FRMSZ)...
; RH(I-M)          PITCH FILTER INPUT #2  ...WHERE FRMSZ FIXED AT 216)
; RH(I-M-1)        PITCH FILTER INPUT #3 (OLDEST)
; QP(I)            N.S. FILTER (ZEROS) OUTPUT: SUM[ANS(J)*Q1(I-J)],J=1:6
; QP1(I)           N.S. FILTER (POLES) OUTPUT: SUM[ AH(J)*Q1(I-J)],J=1:6
; VHP(I)           SPECTRAL FILTER OUTPUT:     SUM[ AH(J)*VH(I-J)],J=1:6
; RHP(I)           PITCH FILTER OUTPUT:  C1H*RH(I-M+1) +
;                                        C2H*RH(I-M)   +
;                                        C3H*RH(I-M-1)
; SP(I)            SPEECH INPUT SAMPLE FOR CURRENT APC LOOP ITERATION (U(I))
; WO(I)            UNQUANTIZED APC RESIDUAL: SP(I)+QP(I)-QP1(I)+VHP(I)+RHP(I)
; GFAC(1:3)        QUANTIZER SCALE FACTORS FOR FRAME SEGMENTS 1:3 (W(0,2,4))
; UO(I)            NORMALIZED APC RESIDUAL: WO/GFAC
; IU(I)            CODED APC RESIDUAL SAMPLE (Y(I))
; UH(I)            QUANTIZED APC RESIDUAL SAMPLE
; WH(I)            SCALED QUANTIZED APC RESIDUAL SAMPLE: UH*GFAC
; Q(I)             CURRENT SAMPLE QUANTIZATION NOISE: WH(I)-WO(I)
;
; THIS APU PROGRAM (IN CONJUNCTION WITH ITS APS PROGRAM)
; MODIFIES ONE INSTRUCTION IN ITS APS PROGRAM BY:
;  0. READING PITCH FROM SCALAR A, AND FIXING IT
;  1. WRITING IT INTO THE RH OF THE BUS-1 COPY OF THE INSTRUCTION
```

```
;  2. WAITING ON DQE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORER
;     SAYS IS NECESSARY)
;  3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
;  4. WRITING IT TO APSMEM IN PSEUDOMEMORY
;  5. WAITING FOR DQE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
;     PROCEED.
;
; GENERAL SCHEME:
;       1) FIX PITCH, SEND TO APS PROGRAM
;       2) INITIALIZE REGISTERS FOR LOOP: GET VH'S, RH'S
;       3) APC LOOP:
;              COMPUTE FILTER OUTPUTS
;              COMPUTE RESIDUAL: WO = SP(I)+QP(I)-QP1(I)+RHP(I)+VHP(I)
;              NORMALIZE RESIDUAL: UO = WO/GFAC
;              CODE/QUANTIZE NORMALIZED RESIDUAL
;              SCALE QUANTIZED RESIDUAL: WH = UH*GFAC
;              UPDATE FILTER INPUTS: Q(I) = WH(I) - WO(I)
;                                    Q1(I)= Q(I) - QP1(I)
;                                    VH(I)= WH(I)- VHP(I)
;                                    RH(I)= VH(I)- RHP(I)
;              DO NEXT APC LOOP ITERATION: I <= I+1
;
; NOTE: SPECTRAL FILTER INPUTS AND PITCH FILTER INPUTS (VH'S & RH'S)
;       ARE RETAINED FROM ITERATION TO ITERATION IN APU REGISTERS.
;       N.S. FILTER INPUTS, AND ALL FILTER COEFFS, ARE REREAD FOR
;       EACH ITERATION OF THE APC LOOP.  N.S. FILTER INPUTS ARE
;       ALSO UPDATED AND WRITTEN OUT AFTER EACH ITERATION'S USE.
;
;
; REGISTER USAGE (FOR MAIN APC LOOP):
;
;       ADM1                                ADM2
;       ----                                ----
; A0: RHP          A4: VH(I-1)          A0: WO SUM/WO/Q A4: VH(I-2)
; A1: RH(I-M)      A5: VH(I-3)          A1: QP1         A5: VH(I-4)
; A2: RH(I-M-1) A6: VH(I-5)             A2: VHP         A6: VH(I-6)
; A3: SCRATCH      A7: SCRATCH          A3: SCRATCH     A7: SCRATCH
;
; M0: ANS(1)/      M4: Q1(I-1)/         M0: ANS(2)/     M4: Q1(I-2)/
;     AH(1)/           VH(I-1)/             AH(2)           VH(I-2)
;     RH(I-M-1)        C3H
; M1: ANS(3)/      M5: Q1(I-3)/         M1: ANS(4)/     M5: Q1(I-4)/
;     AH(3)            VH(I-3)              AH(4)           VH(I-4)
; M2: ANS(5)/      M6: Q1(I-5)/         M2: ANS(6)/     M6: Q1(I-6)/
;     AH(5)            VH(I-5)              AH(6)           VH(I-6)
; M3: RH(I-M)      M7: C2H              M3: WO/         M7: (1/GFAC)/
;                                           UH/             GFAC/
;                                           RH(I-M+1)       C1H
;
;
;
;!!! ALL BUFFERS MUST BE RL,LNG,CONTIGUOUS,
;!!! EXCEPT BUFFER Y WHICH MUST BE FXD,LNG,CNTG.
```

```
             ;
             ; (REQUIRES SPECIAL BINDING TO ACCOMODATE R, S & T BUFFERS)
             ;
             ; INPUT SEQUENCE:
             ;          SYMBOLS                           BUFFER NAMES
             ;          -------                           ------------
             ;          (2**-15), PITCH,                  (2**-15), SA,
             ;          (MOD APS INSTR),                  (MOD APS INSTR),
             ;          RN(-M), RN(-M-1),                 R(FRMSZ-M), R(FRMSZ-M-1),
             ;          VN(-6),...,VN(-1),                S(0),...,S(5),
             ;LOOP FRMSZ (=216) TIMES:
             ; |        SP(I),                            U(I),
             ; |        QI(I-6),ANS(6),                   T(0),V(14),
             ; |             ...,                              ...,
             ; |        QI(I-1),ANS(1),                   T(5),V(9),
             ; |        AN(6),...,AN(1),                  V(5),...,V(0),
             ; |        C1N,C2N,C3N,                      V(6),V(7),V(8),
             ; |        RN(I-M+1),                        R(FRMSZ+I-M+1),
             ; |        (1/GFAC),                         W(1) FOR ITERS 0:71
             ; |                                          W(3) FOR ITERS 72:143
             ; |                                          W(5) FOR ITERS 144:215,
             ; |        (CODING THRESH),                  (CODING THRESH),
             ; |        (2**-15),                         (2**-15),
             ; |        GFAC,                             W(0) FOR ITERS 0:71
             ; |                                          W(2) FOR ITERS 72:143
             ; |                                          W(4) FOR ITERS 144:215,
             ; |        (QUANT VAL(1)),                   (QUANT VAL(1)),
             ; |        (QUANT VAL(2)),                   (QUANT VAL(2)),
             ; |        (QUANT VAL(3)),                   (QUANT VAL(3)),
             ; |        (QUANT VAL(4)),                   (QUANT VAL(4)),
             ;END OF LOOP
             ;          [FI]                              [FI]
             ;
             ;
             ; OUTPUT SEQUENCE:
             ;          SYMBOLS                           BUFFER NAMES
             ;          -------                           ------------
             ;          (FXD PITCH), (MOD APS INSTR),     (FXD PITCH), (MOD APS INSTR),
             ;LOOP FRMSZ (=216) TIMES:
             ; |        QI(I-6),...,QI(I-2),              T(0),...,T(4),
             ; |        IU(I),                            Y(I),
             ; |        QI(I-1),                          T(5),
             ; |        RN(I),                            R(FRMSZ+I),
             ;END OF LOOP
             ;          VN(I-6),...,VN(I-1),              S(0),...,S(5),
             ;          [EO]                              [EO]
             ;
             ;
                        EVEN
06A70 0000              DATA    APCU$SA
06A71 00AB              DATA    APCU$SZ
             ;
             APCU$:     BEGIN   APU(APCU)
```

PAGE 92: [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:38:40, Ed: WOLF
APU3 - APC(Y,A,U,V,W,R,S,T)   APC (WITH NOISE SHAPING)

```
                    APCUSSA:
                    ;
                    ; SEND PITCH TO APS PROGRAM:
                    ;       CONVERT TO FIXED IN LEFT HWORD;
                    ;       WRITE TO BUS1 COPY OF APS PGM;
                    ;       READ RWORD FROM BUS1, WRITE TO APS MEM.;
                    ;
                    ;
                    ; REGISTER USAGE: (FOR APU-APS COMMUNICATION CODE)
                    ;
                    ;       ADM1
                    ;       ----
                    ; A0: FIXED PITCH (IN LEFT HWORD)
                    ; M0: (2**-15)
                    ; M4: PITCH
                    ;
A00 06A72 00E00000          MOV(IQA,M0) \ NOP           (2**-15)
A01 06A74 00EC0000          MOV(IQA,M4) \ NOP           PITCH (REAL)
A02 06A76 04000000          MUL(M0,M4) \ NOP            FIX PITCH
A03 06A78 00B00000          MOV(P,A0) \ NOP
A04 06A7A 3A000000          ALIGN(A0) \ NOP             PUT IN LEFT HWORD
A05 06A7C 009C0000          MOV(R,OQ) \ NOP             OUTPUT FID PITCH TO BUS1
                    ;
A06 06A7E 90100006  #1      JUMPC(#1,OQE)               WAIT FOR WRITE COMPLETION
A07 06A80 00000000          NOP
A08 06A82 20372037          CLEAR(WI)                   LET APS CONTINUE
                    ;
A09 06A84 00FC0000          MOV(IQA,OQ) \ NOP           WRITE FWORD FROM BUS1 TO APS MEM
                    ;
A0A 06A86 9010000A  #2      JUMPC(#2,OQE)               WAIT FOR WRITE COMPLETION
A0B 06A88 00000000          NOP
A0C 06A8A 20372037          CLEAR(WI)                   LET APS CONTINUE
                    ;
                    ;
                    ; INITIALIZE REGISTERS FOR MAIN APS LOOP
                    ; GET VN'S , RN(I-N-1), RN(I-N)
                    ;
A0D 06A8C 00F10000          MOV(IQA,A1) \ NOP           A1 <= RN(I-N)
A0E 06A8E 00F20000          MOV(IQA,A2) \ NOP           A2 <= RN(I-N-1)
                    ;
A0F 06A90 000000F6          NOP \ MOV(IQA,A6)           --- \ A6<=VN(I-6)
A10 06A92 00F60000          MOV(IQA,A6) \ NOP           A6<=VN(I-5) \ ---
A11 06A94 000000F5          NOP \ MOV(IQA,A5)           --- \ A5<=VN(I-4)
A12 06A96 00F50000          MOV(IQA,A5) \ NOP           A5<=VN(I-3) \ ---
A13 06A98 000000F4          NOP \ MOV(IQA,A4)           --- \ A4<=VN(I-2)
A14 06A9A 00F40000          MOV(IQA,A4) \ NOP           A4<=VN(I-1) \ ---
                    ;
                    ;
                    ; MAIN APC LOOP:
                    ;
                    APCUSTOP:
A15 06A9C 000000F0          NOP \ MOV(IQA,A0)           --- \ A0<=SP(I)
                    ;
```

```
                       (04086) ;
                       (04087) ; COMPUTE PREDICTIONS FOR SAMPLE I; COMPUTE APC RESIDUAL, WO:
                       (04088) ; --------------------------------  ----------------------
                       (04089) ;
                       (04090) ;
                       (04091) ; READ IN INPUTS AND COEFFS FOR NOISE SHAPING FILTER (ZEROS)
                       (04092) ; READ PAIR OF INPUTS & COEFFS, START MULT, READ NEXT PAIR.
                       (04093) ; Q1(I-6),ANS(6), Q1(I-5),ANS(5), ETC...
                       (04094) ; WRITE Q1(I-5),...,Q1(I-1) BACK OUT AS NEXT ITERATION'S
                       (04095) ; Q1(I-6),...,Q1(I-2) AS THEY ARE READ IN.
                       (04096) ; (NEXT ITER'S Q1(I-1) WILL BE CALCULATED & WRITTEN LATER AS
                       (04097) ;  CURRENT Q1(I))
                       (04098) ;
A16 06A9E 000000EE (04099)       NOP \ MOV(IQA,M6)              --- \ M6<=Q1(I-6)
A17 06AA0 000000EA (04100)       NOP \ MOV(IQA,M2)              --- \ M2<=ANS(6)
A18 06AA2 00EE00DC (04101)       MOV(IQA,M6) \ MOV(IQ,DQ)       M6<=Q1(I-5) \ DQ<=Q1(I-5)
A19 06AA4 00EA0000 (04102)       MOV(IQA,M2) \ NOP              M2<=ANS(5) \ ---
A1A 06AA6 05400540 (04103)       MUL(M2,M6)                     P<=ANS(5)Q1(I-5) \ P<=ANS(6)Q1(I-6)
                       (04104) ;
A1B 06AA8 00DC00ED (04105)       MOV(IQ,DQ) \ MOV(IQA,M5)       DQ<=Q1(I-4) \ M5<=Q1(I-4)
A1C 06AAA 000000E9 (04106)       NOP \ MOV(IQA,M1)              --- \ M1<=ANS(4)
A1D 06AAC 00ED00DC (04107)       MOV(IQA,M5) \ MOV(IQ,DQ)       M5<=Q1(I-3) \ DQ<=Q1(I-3)
A1E 06AAE 00E90000 (04108)       MOV(IQA,M1) \ NOP              M1<=ANS(3) \ ---
A1F 06AB0 84038483 (04109)       MOV(A3),MUL(M1,M5)             A3<=FIRST N.S. ZEROS PRODUCTS
                       (04110) ;                                P<=ANS(3)Q1(I-3) \ P<=ANS(4)Q1(I-4)
                       (04111) ;
A20 06AB2 00DC00EC (04112)       MOV(IQ,DQ) \ MOV(IQA,M4)       DQ<=Q1(I-2) \ M4<=Q1(I-2)
A21 06AB4 000000E8 (04113)       NOP \ MOV(IQA,M0)              --- \ M0<=ANS(2)
A22 06AB6 00EC00DC (04114)       MOV(IQA,M4) \ MOV(IQ,DQ)       M4<=Q1(I-1) \ DQ<=Q1(I-1)
A23 06AB8 00E80000 (04115)       MOV(IQA,M0) \ NOP              M0<=ANS(1) \ ---
A24 06ABA 04170417 (04116)       MOV(A7),MUL(M0,M4)             A7<=MIDDLE N.S. ZEROS PRODUCTS
                       (04117) ;                                P<=ANS(1)Q1(I-1) \ P<=ANS(2)Q1(I-2)
                       (04118) ;
                       (04119) ; ADD PRODUCTS SO FAR...
                       (04120) ;
A25 06ABC 47604760 (04121)       ADD(A3,A7)                     R<=SUM(1ST 2 SETS OF N.S.ZEROS PRODS)
                       (04122) ;
                       (04123) ; START MOVING IN COEFFS FOR N.S. POLES & SPECTRAL FILTER
                       (04124) ; AN(6),...,AN(1)
                       (04125) ;
A26 06ABE 000000EA (04126)       NOP \ MOV(IQA,M2)              --- \ M2<=AN(6)
A27 06AC0 00EA0000 (04127)       MOV(IQA,M2) \ NOP              M2<=AN(5) \ ---
                       (04128) ;
                       (04129) ; MOVE LAST N.S. ZEROS PRODUCT TO ADDER
                       (04130) ; START MULT FOR N.S. POLES
                       (04131) ;
A28 06AC2 85570557 (04132)       MOV(A7),MUL(M2,M6)             A7<=LAST N.S. ZEROS PRODUCT
                       (04133) ;                                P<=AN(5)Q1(I-5) \ P<=AN(6)Q1(I-6)
                       (04134) ;
                       (04135) ; ADD LAST N.S. ZEROS PRODUCT TO ACCUM
                       (04136) ;
A29 06AC4 47734773 (04137)       MOV(A3),ADD(A3,A7)             A3<=SUM(1ST 2 SETS OF N.S.ZEROS PRODS)
                       (04138) ;                                R<=SUM(ALL 3 SETS OF N.S.ZEROS PRODS)
```

```
                        (04139) ;
                        (04140) ; GET NEXT 2 AN'S
                        (04141) ;
A2A 06AC6 000000E9 (04142)        NOP \ MOV(IQA,M1)                --- \ M1<=AN(4)
A2B 06AC8 00E90000 (04143)        MOV(IQA,M1) \ NOP                M1<=AN(3) \ ---
                        (04144) ;
                        (04145) ; GET 2 N.S. ZEROS PARTIAL ACCUMS (PARTIAL QP'S) INTO ADM2,
                        (04146) ; ADD THEM, ADD RESULT TO WO ACCUM.
                        (04147) ;
A2C 06ACA 00900093 (04148)        MOV(R,EXO) \ MOV(R,A3)           EXO<=PARTIAL QP \ A3<=PARTIAL QP
A2D 06ACC 00000057 (04149)        NOP \ MOV(EXI,A7)                --- \ A7<=PARTIAL QP
A2E 06ACE 00004760 (04150)        NOP \ ADD(A3,A7)                 --- \ R<=QP(I)
                        (04151) ;
                        (04152) ; DONE WITH ZEROS (STILL HAVE TO ADD QP(I) TO A0 (WO ACCUM))
                        (04153) ; CONTINUE WITH POLES...
                        (04154) ;
A2F 06AD0 000000E8 (04155)        NOP \ MOV(IQA,M0)                --- \ M0<=AN(2)
A30 06AD2 00E80000 (04156)        MOV(IQA,M0) \ NOP                M0<=AN(1) \ ---
A31 06AD4 04030403 (04157)        MOV(A3),MUL(M1,M5)               A3<=1ST N.S. POLES PRODUCTS
                        (04158) ;                                  P<=AN(3)Q1(I-3) \ P<=AN(4)Q1(I-4)
                        (04159) ;
A32 06AD6 00004717 (04160)        NOP \ MOV(A7),ADD(A0,A7)         --- \ A7<=QP(I) , R<=SP(I)+QP(I)
A33 06AD8 00000000 (04161)        NOP
A34 06ADA 00000000 (04162)        NOP
A35 06ADC 04170417 (04163)        MOV(A7),MUL(M0,M4)               A7<=MIDDLE N.S. POLES PRODUCTS
                        (04164) ;                                  P<=AN(1)Q1(I-1) \ P<=AN(2)Q1(I-2)
                        (04165) ;
A36 06ADE 47604770 (04166)        ADD(A3,A7) \ MOV(A0),ADD(A3,A7) --- \ A0<=SP(I)+QP(I)
                        (04167) ;                                  R<=SUM(1ST 2 SETS N.S.POLES PRODS)
                        (04168) ;
                        (04169) ; MEANWHILE, GET 1ST 2 PITCH FILTER COEFFS...
                        (04170) ;
A37 06AE0 000000EF (04171)        NOP \ MOV(IQA,M7)                --- \ M7<=C1N
A38 06AE2 00EF0000 (04172)        MOV(IQA,M7) \ NOP                M7<=C2N \ ---
                        (04173) ;
A39 06AE4 00870087 (04174)        MOV(P,A7)                        A7<=LAST N.S. POLES PRODUCTS
A3A 06AE6 47734773 (04175)        MOV(A3),ADD(A3,A7)               A3<=SUM(1ST 2 SETS N.S.POLES PRODS)
                        (04176) ;                                  R<=SUM(ALL 3 SETS N.S.POLES PRODS)
                        (04177) ;
                        (04178) ; GET 2 N.S.POLES PARTIAL ACCUMS (PARTIAL QP1'S) INTO ADM2,
                        (04179) ; ADD THEM, SAVE IN A1, SUBTRACT RESULT FROM WO ACCUM.
                        (04180) ;
A3B 06AE8 00900093 (04181)        MOV(R,EXO) \ MOV(R,A3)           EXO<=PARTIAL QP1 \ A3<=PARTIAL QP1
A3C 06AEA 00000057 (04182)        NOP \ MOV(EXI,A7)                --- \ A7<=PARTIAL QP1
A3D 06AEC 00004760 (04183)        NOP \ ADD(A3,A7)                 --- \ P<=QP1(I)
A3E 06AEE 00004911 (04184)        NOP \ MOV(A1),SUB(A0,A1)         --- \ A1<=QP1(I);R<=SP(I)+QP(I)-QP1(I)
                        (04185) ;
                        (04186) ; START MOVING VN'S (SPECTRAL FILTER INPUTS) FROM A4,A5,A6 TO M4,M5,M6
                        (04187) ; AT SAME TIME, UPDATE VN'S IN A4,A5,A6 FOR NEXT ITERATION:
                        (04188) ;  CURRENT VN(I-N) => NEXT ITER'S VN(I-(N+1))
                        (04189) ;
A3F 06AF0 02000290 (04190)        R(A4) \ MOV(A0),R(A4)            R<=VN(I-1) \ R<=VN(I-2);A0<=SP+QP-QP1
A40 06AF2 00000000 (04191)        NOP
```

```
PAGE   95:  [BBND]<DCA16>BBN16N.MSO.2, 29-Dec-80 14:38:48, Ed: WOLF
             APU3 - APC(Y,A,U,V,W,R,S,T)    APC (WITH NOISE SHAPING)

 A41 06AF4 00000000 (04192)          NOP
 A42 06AF6 08980898 (04193)          MOV(R,EXO)                    EXO<=VH(I-1) \ EXO<=VH(I-2)
 A43 06AF8 02AC02AC (04194)          MOV(M4),R(A5)                 M4<=VH(I-1) \ M4<=VH(I-2)
                    (04195) ;                                      R <=VH(I-3) \ R <=VH(I-4)
                    (04196) ;
                    (04197) ; START SPECTRAL FILTER MULTS
                    (04198) ;
 A44 06AFA 04000400 (04199)          MUL(M0,M4)                    P<=AH(1)VH(I-1) \ P<=AH(2)VH(I-2)
 A45 06AFC 00000000 (04200)          NOP
 A46 06AFE 00550054 (04201)          MOV(EXI,A5) \ MOV(EXI,A4)     A5<=VH(I-2) (=NEXT ITER VH(I-3))
                    (04202) ;                                        \ A4<=VH(I-1) (=NEXT ITER VH(I-2))
 A47 06B00 08980898 (04203)          MOV(R,EXO)                    EXO<=VH(I-3) \ EXO<=VH(I-4)
 A48 06B02 02CD02CD (04204)          MOV(M5),R(A6)                 M5<=VH(I-3) \ M5<=VH(I-4)
                    (04205) ;                                      R <=VH(I-5) \ R <=VH(I-6)
                    (04206) ;
 A49 06B04 04830483 (04207)          MOV(A3),MUL(M1,M5)            A3<=1ST SPECTRAL PRODUCTS
                    (04208) ;                                      P<=AH(3)VH(I-3) \ P<=AH(4)VH(I-4)
                    (04209) ;
                    (04210) ; WHILE WAITING FOR MULT, GET LAST PITCH FILTER COEFF
                    (04211) ;
 A4A 06B06 00EC0000 (04212)          MOV(IQA,M4) \ NOP             M4<=C3H \ ---
                    (04213) ;
 A4B 06B08 00560055 (04214)          MOV(EXI,A6) \ MOV(EXI,A5)     A6<=NEXT VH(I-5) \ A5<=NEXT VH(I-4)
 A4C 06B0A 08980898 (04215)          MOV(R,EXO)                    EXO<=VH(I-5) \ EXO<=VH(I-6)
 A4D 06B0C 00000056 (04216)          NOP \ MOV(EXI,A6)             TOSS OLD VH(I-6) \ A6<=NEXT VH(I-6)
 A4E 06B0E 024E080E (04217)          MOV(M6),R(A2) \ MOV(R,M6)     M6<=VH(I-5);R<=RH(I-M-1) \ M6<=VH(I-6)
                    (04218) ;
 A4F 06B10 05570557 (04219)          MOV(A7),MUL(M2,M6)            A7<=MIDDLE SPECTRAL PRODUCTS
                    (04220) ;                                      P<=AH(5)VH(I-5) \ P<=AH(6)VH(I-6)
                    (04221) ;
 A50 06B12 00000000 (04222)          NOP
 A51 06B14 022R0000 (04223)          MOV(M0),R(A1) \ NOP           M0<=RH(I-M-1);R<=RH(I-M) \ ---
 A52 06B16 00000000 (04224)          NOP
 A53 06B18 00000000 (04225)          NOP
 A54 06B1A 00F100CB (04226)          MOV(IQA,A1) \ MOV(IQ,M3)      A1<=RH(I-M+1) (=NEXT ITER RH(I-M)) \
                    (04227) ;                                      M3<=RH(I-M+1)
 A55 06B1C 00920000 (04228)          MOV(R,A2) \ NOP               A2<=NEXT RH(I-M-1)
 A56 06B1E 476B4760 (04229)          MOV(M3),ADD(A3,A7) \ ADD(A3,A7) M3<=RH(I-M) \ ---
                    (04230) ;                                      R<=SUM(1ST 2 SETS OF SPECTRAL PRODUCTS)
 A57 06B20 00000000 (04231)          NOP
 A58 06B22 00000000 (04232)          NOP
                    (04233) ;
                    (04234) ; START PITCH FILTER MULTS
                    (04235) ;
 A59 06B24 05F705F7 (04236)          MOV(A7),MUL(M3,M7)            A7<=LAST SPECTRAL PRODUCTS
                    (04237) ;                                      P<=RH(I-M)C2H \ P<=RH(I-M+1)C1H
 A5A 06B26 47734773 (04238)          MOV(A3),ADD(A3,A7)            A3<=SUM(1ST 2 SETS OF SPECTRAL PRODS)
                    (04239) ;                                      R<=SUM(ALL 3 SETS OF SPECTRAL PRODS)
                    (04240) ;
                    (04241) ; GET 2 SPECTRAL PARTIAL ACCUMS (PARTIAL VHP'S) INTO ADM2,
                    (04242) ; ADD THEM, SAVE IN A2, ADD RESULT TO WO ACCUM.
                    (04243) ;
 A5B 06B28 08980893 (04244)          MOV(R,EXO) \ MOV(R,A3)        EXO<=PARTIAL VHP \ A3<=PARTIAL VHP
```

```
PAGE   96:  [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
              APU3 - APC(Y,A,U,V,W,R,S,T)    APC (WITH NOISE SHAPING)

 A5C 06B2A 00000857 (04245)          NOP \ MOV(EXI,A7)                ---- \ A7<=PARTIAL VHP
 A5D 06B2C 00004760 (04246)          NOP \ ADD(A3,A7)                 ---- \ R<=VHP(I)
 A5E 06B2E 00004212 (04247)          NOP \ MOV(A2),ADD(A0,A2)         ---- \ A2<=VHP(I);R<=SP+QP-QP1+VHP
                    (04248) ;
                    (04249) ; CONTINUE WITH PITCH FILTER..
                    (04250) ;
 A5F 06B30 84130000 (04251)          MOV(A3),MUL(M0,M4) \ MOV(P,EXO) A3<=RN(I-M)C2N \ EXO<=RN(I-M+1)C1N
                    (04252) ;                                         P<=RN(I-M-1)C3N \ ---
 A60 06B32 00570000 (04253)          MOV(EXI,A7) \ NOP                A7<=RN(I-M+1)C1N \ ---
 A61 06B34 47600890 (04254)          ADD(A3,A7) \ MOV(R,A0)           R<=SUM(1ST 2 PITCH PRODS) \
                    (04255) ;                                           A0<=SP(I)+QP(I)-QP1(I)+VHP(I)
                    (04256) ;
                    (04257) ; START SET-UP FOR WO NORMALIZATION
                    (04258) ;
 A62 06B36 000000EF (04259)          NOP \ MOV(IQA,M7)                ---- \ M7<=(1/GFAC)
 A63 06B38 00000000 (04260)          NOP
 A64 06B3A 00000000 (04261)          NOP
                    (04262) ;
 A65 06B3C 00870000 (04263)          MOV(P,A7) \ NOP                  A7<=RN(I-M-1)C3N \ ---
 A66 06B3E 47730000 (04264)          MOV(A3),ADD(A3,A7) \ NOP         A3<=SUM(1ST 2 PITCH PRODS);R<=RMP(I)\--
 A67 06B40 00000000 (04265)          NOP
 A68 06B42 00000000 (04266)          NOP
 A69 06B44 00980000 (04267)          MOV(R,EXO) \ NOP                 EXO<=RMP(I) \ ---
 A6A 06B46 00900857 (04268)          MOV(R,A0) \ MOV(EXI,A7)          A0<=RMP(I) \ A7<=RMP(I)
 A6B 06B48 00004700 (04269)          NOP \ ADD(A0,A7)                 ---- \ R<=SP+QP-QP1+VHP+RMP (=WO)
 A6C 06B4A 00000000 (04270)          NOP
 A6D 06B4C 00000000 (04271)          NOP
 A6E 06B4E 00000890 (04272)          NOP \ MOV(R,A0)                  ---- \ A0<=WO
 A6F 06B50 0000080B (04273)          NOP \ MOV(R,M3)                  ---- \ M3<=WO
                    (04274) ;
                    (04275) ;
                    (04276) ; NORMALIZE APC RESIDUAL:
                    (04277) ; ---------------------
                    (04278) ;
                    (04279) ;
                    (04280) ; CALCULATE UO (NORMALIZED RESIDUAL)
                    (04281) ; BY MULTIPLYING WO BY (1/GFAC)
                    (04282) ;
 A70 06B52 000085E0 (04283)          NOP \ MUL(M3,M7)                 ---- \ P<=WO(1/GFAC)
                    (04284) ;
                    (04285) ;
                    (04286) ; CODE-DECODE NORMALIZED RESIDUAL:
                    (04287) ; ------------------------------
                    (04288) ;
                    (04289) ;
                    (04290) ; CODE AND QUANTIZE UO, GENERATING IU (CODED) & UH (QUANTIZED)
                    (04291) ;
                    (04292) ; CODING TABLE (FOLDED BINARY CODE): (DON'T WORRY ABOUT BOUNDARY CONDITIONS)
                    (04293) ; BIN#           UO RANGE         IU       UH
                    (04294) ; ----           --------         --       --
                    (04295) ;  1             UO < -THRESH      1       VAL(1)
                    (04296) ;  2     -THRESH < UO < 0          0       VAL(2)
                    (04297) ;  3           0 < UO <  THRESH    2       VAL(3)
```

```
                         (04298) ;  4     THRESH < UO           3       VAL(4)
                         (04299) ;
A71 06854 000008F7 (04300)          NOP \ MOV(IQA,A7)                   --- \ A7<=THRESH
                         (04301) ;
                         (04302) ; WHILE WAITING ON MULT, GET CODED 1 & 2 INTO ADM1'S A7 & A3
                         (04303) ; ADD THEM, LEAVE CODED 3 IN ADM1'S R REGISTER
                         (04304) ;
A72 06856 00870000 (04305)          MOV(IQ,A7) \ NOP                    A7<=(2**-15) (FXD 1) \--
A73 06858 00F30000 (04306)          MOV(IQA,A3) \ NOP                   A3<=(FXD 1) \ ---
A74 0685A 47600000 (04307)          ADD(A3,A7) \ NOP                    R<=(FXD 2) \ ---
A75 0685C 00000000 (04308)          NOP
                         (04309) ;
A76 0685E 000000B3 (04310)          NOP \ MOV(P,A3)                     --- \ A3<=UO
A77 06860 47730260 (04311)          MOV(A3),ADD(A3,A7) \ R(A3)          A3<=(FXD 2);R<=(FXD 3) \ R<=UO
                         (04312) ;                                      --- \ T2<=SIGN(UO)
A78 06862 000008EF (04313)          NOP \ MOV(IQA,M7)                   --- \ M7<=GFAC
A79 06864 00000080 (04314)          NOP \ MOV(R,NULL)                   --- \ WAIT FOR COMPLETION
A7A 06866 911F0089 (04315)          JUMPS(APCU$B12,T2)                  JUMP IF UO NEGATIVE (BINS 1&2)
                         (04316) ;
                         (04317) ; HERE IF UO ZERO OR POSITIVE (BINS 3&4)
                         (04318) ;
                         (04319) ;APCU$B34:
A7B 06868 00004F60 (04320)          NOP \ SUB(A3,A7)                    --- \ R<=UO-THRESH
A7C 0686A 00000080 (04321)          NOP \ MOV(R,NULL)                   --- \ T2<=0 IF (UO > THRESH)
                         (04322) ;                                      --- \ T2<=1 IF (UO < THRESH)
A7D 0686C 911F0084 (04323)          JUMPS(APCU$B3,T2)                   JUMP IF (0 < UO < THRESH) (BIN 3)
                         (04324) ;
                         (04325) ; HERE FOR BIN 4
                         (04326) ;
                         (04327) ;APCU$B4:
A7E 0686E 009300E0 (04328)          MOV(R,A3) \ MOV(IQA,NULL)           A3<=(FXD 3)(=IU) \ TOSS VAL(1)
A7F 06870 3A6000E0 (04329)          ALIGN(A3) \ MOV(IQA,NULL)           R<=(FXD 3) \ TOSS VAL(2)
A80 06872 000000E0 (04330)          NOP \ MOV(IQA,NULL)                 --- \ TOSS VAL(3)
A81 06874 000000EB (04331)          NOP \ MOV(IQA,M3)                   --- \ M3<=VAL(4) (=UH)
A82 06876 009C0000 (04332)          MOV(R,DQ) \ NOP                     DQ<=(FXD 3) \ ---
A83 06878 10000095 (04333)          JUMP(APCU$COM)                      JUMP TO COMMON CODE
                         (04334) ;
                         (04335) ; HERE FOR BIN 3
                         (04336) ;
                         (04337) APCU$B3:
A84 0687A 3A6000E0 (04338)          ALIGN(A3) \ MOV(IQA,NULL)           R<=(FXD 2)(=IU) \ TOSS VAL(1)
A85 0687C 000000E0 (04339)          NOP \ MOV(IQA,NULL)                 --- \ TOSS VAL(2)
A86 0687E 000000EB (04340)          NOP \ MOV(IQA,M3)                   --- \ M3<=VAL(3) (=UH)
A87 06880 009C00E0 (04341)          MOV(R,DQ) \ MOV(IQA,NULL)           DQ<=IU \ TOSS VAL(4)
A88 06882 10000095 (04342)          JUMP(APCU$COM)                      JUMP TO COMMON CODE
                         (04343) ;
                         (04344) ; HERE IF UO NEGATIVE (BINS 1&2)
                         (04345) ;
                         (04346) APCU$B12:
A89 06884 3AE04760 (04347)          ALIGN(A7) \ ADD(A3,A7)              R<=(FXD 1) \ R<=UO+THRESH
A8A 06886 00000080 (04348)          NOP \ MOV(R,NULL)                   --- \ T2<=0 IF (UO > -THRESH)
                         (04349) ;                                      --- \ T2<=1 IF (UO < -THRESH)
A8B 06888 911F0091 (04350)          JUMPS(APCU$B1,T2)                   JUMP IF (UO < -THRESH) (BIN 1)
```

```
                        (04351) ;
                        (04352) ; HERE FOR BIN 2
                        (04353) ;
                        (04354) ;APCU$B2:
A8C 06B8A 001C08E0 (04355)         MOV(ZERO,DQ) \ MOV(IQA,NULL)    DQ<=(FXD 0)(=IU) \ TOSS VAL(1)
A8D 06B8C 000008E0 (04356)         NOP \ MOV(IQA,M3)               --- \ M3<=VAL(2) (=UH)
A8E 06B8E 000008E0 (04357)         NOP \ MOV(IQA,NULL)             --- \ TOSS VAL(3)
A8F 06B90 000008E0 (04358)         NOP \ MOV(IQA,NULL)             --- \ TOSS VAL(4)
A90 06B92 10000095 (04359)         JUMP(APCU$COM)                  JUMP TO COMMON CODE
                        (04360) ;
                        (04361) ; HERE FOR BIN 1
                        (04362) ;
                        (04363) APCU$B1:
A91 06B94 009C08E0 (04364)         MOV(R,DQ) \ MOV(IQA,M3)         DQ<=(FXD 1)(=IU) \ M3<=VAL(1)(=UH)
A92 06B96 000008E0 (04365)         NOP \ MOV(IQA,NULL)             --- \ TOSS VAL(2)
A93 06B98 000008E0 (04366)         NOP \ MOV(IQA,NULL)             --- \ TOSS VAL(3)
A94 06B9A 000008E0 (04367)         NOP \ MOV(IQA,NULL)             --- \ TOSS VAL(4)
                        (04368) ;
                        (04369) ;
                        (04370) ; HERE WHEN DONE WITH CODING AND QUANTIZATION
                        (04371) ; UH IS IN M3.
                        (04372) ;
                        (04373) APCU$COM:
                        (04374) ;
                        (04375) ;
                        (04376) ; SCALE THE QUANTIZED RESIDUAL:
                        (04377) ; ----------------------------
                        (04378) ;
A95 06B9C 000085E0 (04379)         NOP \ MUL(M3,M7)                --- \ P<=(UH)(GFAC) (=WH)
                        (04380) ;
                        (04381) ;
                        (04382) ; UPDATE ARRAYS:
                        (04383) ; -------------
                        (04384) ;
A96 06B9E 000008B7 (04385)         NOP \ MOV(P,A7)                 --- \ A7<=WH
A97 06BA0 000048E0 (04386)         NOP \ SUB(A7,A0)                --- \ R<=WH-WO (=Q)
A98 06BA2 00004910 (04387)         NOP \ MOV(A0),SUB(A0,A1)        --- \ A0<=Q(I);R<=Q-QP1 (=NEXT Q1(I-1))
A99 06BA4 00004AFC (04388)         NOP \ MOV(DQ),SUB(A7,A2)        --- \ DQ<=NEXT Q1(I-1);
                        (04389) ;                                  --- \ R<=WH-VHP (=NEXT VH(I-1))
A9A 06BA6 00000898 (04390)         NOP \ MOV(R,EXO)                --- \ EXO<=NEXT VH(I-1)
A9B 06BA8 08540000 (04391)         MOV(EXI,A4) \ NOP               A4<=NEXT VH(I-1) \ ---
A9C 06BAA 40800000 (04392)         SUB(A4,A0) \ NOP                R<=(NEXT VH(I-1) - RHP) (=RH)
A9D 06BAC 009C0000 (04393)         MOV(R,DQ) \ NOP                 DQ<=RH(I) \ ---
                        (04394) ;
                        (04395) ;
                        (04396) ; DONE WITH APC LOOP.
                        (04397) ; LOOP TIL [FI]
                        (04398) ;
A9E 06BAE 901D0015 (04399)         JUMPC(APCU$TOP,FI)              LOOP
A9F 06BB0 00000000 (04400)         NOP
                        (04401) ;
                        (04402) ;
                        (04403) ; WHEN DONE, WRITE OUT VH'S
```

```
PAGE    99:  [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
               APU3 - APC(Y,A,U,V,W,R,S,T)    APC (WITH NOISE SHAPING)

                           (04404) ; VH(I-6),...,VH(I-1)
                           (04405) ;
  AA0 06BB2 02C002C0 (04406)            R(A6)                          R<=VH(I-5) \ R<=VH(I-6)
  AA1 06BB4 0000089C (04407)            NOP \ MOV(R,DQ)                --- \ DQ<=VH(I-6)
  AA2 06BB6 089C0000 (04408)            MOV(R,DQ) \ NOP                DQ<=VH(I-5) \ ---
  AA3 06BB8 02A002A0 (04409)            R(A5)                          R<=VH(I-3) \ R<=VH(I-4)
  AA4 06BBA 0000089C (04410)            NOP \ MOV(R,DQ)                --- \ DQ<=VH(I-4)
  AA5 06BBC 089C0000 (04411)            MOV(R,DQ) \ NOP                DQ<=VH(I-3) \ ---
  AA6 06BBE 02800280 (04412)            R(A4)                          R<=VH(I-1) \ R<=VH(I-2)
  AA7 06BC0 0000089C (04413)            NOP \ MOV(R,DQ)                --- \ DQ<=VH(I-2)
  AA8 06BC2 089C0000 (04414)            MOV(R,DQ) \ NOP                DQ<=VH(I-1) \ ---
                           (04415) ;
                           (04416) ;
  AA9 06BC4 20322032 (04417)            CLEAR(RA)
  AAA 06BC6 10000000 (04418)            JUMP(0)
            000000AB (04419) APCU$SZ = $A-APCU$SA
      06BC8          (04420)            END
```

```
' APS3 - APC(Y,A,U,V,W,R,S,T)   APC (WITH NOISE SHAPING)
; KFIELD 8/8Ø
;
; BINDING DONE BY SBM$APC TO ACCOMODATE R, S & T BUFFERS
;
;!!! ALL BUFFERS MUST BE RL,LNG,CONTIGUOUS,
;!!! EXCEPT BUFFER Y WHICH MUST BE FXD,LNG,CNTG.
;
; THIS APS PROGRAM (IN CONJUNCTION WITH ITS APU PROGRAM)
; MODIFIES ONE INSTRUCTION IN ITSELF BY:
;  Ø. READING PITCH FROM SCALAR A, AND FIXING IT
;  1. WRITING IT INTO THE RH OF THE BUS-1 COPY OF THE INSTRUCTION
;  2. WAITING ON OQE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORER
;      SAYS IS NECESSARY)
;  3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
;  4. WRITING IT TO APSMEM IN PSEUDOMEMORY
;  5. WAITING FOR OQE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
;      PROCEED.
;
; INPUT SEQUENCE:
;        SYMBOLS                             BUFFER NAMES
;        -------                             ------------
;        (2**-15), PITCH,                    (2**-15), SA,
;        (MOD APS INSTR),                    (MOD APS INSTR),
;        RH(-M), RH(-M-1),                   R(FRMSZ-M), R(FRMSZ-M-1),
;        VH(-6),...,VH(-1),                  S(Ø),...,S(5),
;LOOP FRMSZ (=216) TIMES:
; |      SP(I),                              U(I),
Ø |      Q1(I-6),ANS(6),                     T(Ø),V(14),
; |           ...,                                ...,
; |      Q1(I-1),ANS(1),                     T(5),V(9),
; |      AH(6),...,AH(1),                    V(5),...,V(Ø),
; |      C1H,C2H,C3H,                        V(6),V(7),V(8),
; |      RH(I-M+1),                          R(FRMSZ+I-M+1),
; |      (1/GFAC),                           W(1) FOR ITERS Ø:71
; |                                          W(3) FOR ITERS 72:143
; |                                          W(5) FOR ITERS 144:215,
; |      (CODING THRESH),                    (CODING THRESH),
; |      (2**-15),                           (2**-15),
; |      GFAC,                               W(Ø) FOR ITERS Ø:71
; |                                          W(2) FOR ITERS 72:143
; |                                          W(4) FOR ITERS 144:215,
; |      (QUANT VAL(1)),                     (QUANT VAL(1)),
; |      (QUANT VAL(2)),                     (QUANT VAL(2)),
; |      (QUANT VAL(3)),                     (QUANT VAL(3)),
; |      (QUANT VAL(4)),                     (QUANT VAL(4)),
;END OF LOOP
;        [FI]                                [FI]
;
;
; OUTPUT SEQUENCE:
;        SYMBOLS                             BUFFER NAMES
;        -------                             ------------
```

```
                         ;        (FXD PITCH), (MOD APS INSTR),    (FXD PITCH), (MOD APS INSTR),
                         ;LOOP FRMSZ (=216) TIMES:
                         ; |      Q1(I-6),...,Q1(I-2),             T(0),...,T(4),
                         ; |      IU(I),                           V(I),
                         ; |      Q1(I-1),                         T(5),
                         ; |      RH(I),                           R(FRMSZ+I),
                         ;END OF LOOP
                         ;        VH(I-6),...,VH(I-1),             S(0),...,S(5),
                         ;        [EO]                             [EO]
                         ;
                         ;
                         ; DEFINE SYMBOLIC FRAMESIZE
                         ;
             000000D8             FRMSZ = 216
                         ;
                                  EVEN
       06BC8 00000000             ADDR     0      ;(PTR TO CONS INS BLOCK)
       06BCA 00000000             ADDR     0      ;(SCALAR ADDRESS)
       06BCC 0000                 DATA     0      ;(NUMBER OF SCALARS)
       06BCD 00C8                 DATA     APCS$Z ;MODULE SIZE
       06BCE 00000000             ADDR     APCS$C ;PTR TO CHAIN ANCHOR
                         ;
                         APCS$    BEGIN    APS(APCS)
A00 06BD0 00204A40                JSN(APCS$O,P2)  ;SET UP AND START OUTPUT PROGRAM
A01 06BD2 02300038                SET(RO)
                         ;
                         ; INPUT PROGRAM
                         ;        BR0: SCRATCH
                         ;        BR1: PITCH / R(FRMSZ+I-M+1)
                         ;        BR2: U(I)
                         ;        BR3: INPUT LOOP ITERATION COUNTER ((FRMSZ-1):0)
                         ;
                         APCS$I:
                         ;
                         ;
                         ; DO INSTRUCTION MODIFICATION STUFF:
                         ;
A02 06BD4 04C203CE                LOAD(BR0,SVT$UNI(1),TF)          GEN (2**-15) ADDR
                         APCS$SA:
A03 06BD6 06C20000                LOAD(BR0,M$S(1),TF)              GEN SA ADDR
A04 06BD8 00300037                SET(WI)                          WAIT FOR APU TO SIGNAL
                         ;                                         WRITE COMPLETE
A05 06BDA 0A000020                NOP
                         ;
A06 06BDC 0CC26BE6                LOAD(BR0,APCS$MOD(1),TF)         GEN MOD INSTR ADDR
A07 06BDE 0E300037                SET(WI)                          WAIT FOR WRITE COMPLETE
A08 06BE0 10000020                NOP
                         ;
                         ;
                         ; GEN R(FRMSZ-M), R(FRMSZ-M-1)
                         ; ALSO GET INITIAL R(FRMSZ+I-M+1)  (MINUS SPACING)
                         ;
                         APCS$IR:
```

```
PAGE  102:  [BBND]<DCA16>BBN16M.MSU.2, 29-Dec-80 14:30:40, Ed: WOLF
              APS3 - APC(Y,A,U,V,W,R,S,T)   APC (WITH NOISE SHAPING)

  A09 06BE2 12400000 (04527)          LOAD(BR0,M$$)                  BR0 <= ADDR R(0)
  A0A 06BE4 140A0100 (04528)          ADD(BR0,FRMSZ*W$)              BR0 <= ADDR R(FRMSZ)
                     (04529) ;
                     (04530) ;!!! NEXT INSTRUCTION MODIFIED AT RUNTIME !!!
                     (04531) ;
            00006BE6 (04532) APCS$MOD = @L
  A0B 06BF6 16500000 (04533)          LOAD(BR1,M$$)                  BR1 <= M (PITCH LAG)
                     (04534) ;
  A0C 06BF0 1819002B (04535)          ADDB(BR1,BR1)                  BR1 <= 2*M (FOR SI=2)
                     (04536) ;
  A0D 06BFA 1A090023 (04537)          SUBB(BR0,BR1,TF)               GEN ADDR R(FRMSZ-M)
  A0E 06BFC 1C190011 (04538)          MOVB(BR1,BR0)                  BR1 <= ADDR R(FRMSZ+I-M+1) FOR I=-1
  A0F 06BEE 1E670002 (04539)          SUB(BR0,W$,TF)                 GEN ADDR R(FRMSZ-M-1)
                     (04540) ;
                     (04541) ; GEN S(0),...,S(5)
                     (04542) ;
                     (04543) APCS$IS:
  A10 06BF0 20C00000 (04544)          LOAD(BR0,M$$,TF)               GEN ADDR S(0)
  A11 06BF2 220A0002 (04545)          ADD(BR0,W$,TF)                 GEN ADDR S(1)
  A12 06BF4 240A0002 (04546)          ADD(BR0,W$,TF)                 GEN ADDR S(2)
  A13 06BF6 260A0002 (04547)          ADD(BR0,W$,TF)                 GEN ADDR S(3)
  A14 06BF8 280A0002 (04548)          ADD(BR0,W$,TF)                 GEN ADDR S(4)
  A15 06BFA 2A0A0002 (04549)          ADD(BR0,W$,TF)                 GEN ADDR S(5)
                     (04550) ;
                     (04551) ; INITIALIZE INPUT LOOP:
                     (04552) ; KEEP ITERATION COUNTER IN BR3 ((FRMSZ-1):0)
                     (04553) ;
  A16 06BFC 2C7000D7 (04554)          LOAD(BR3,FRMSZ-1)              BR3 <= FRMSZ-1
                     (04555) ;
                     (04556) ; GET INITAL U BUFFER ADDRESS (MINUS SPACING)
                     (04557) ; (INITIAL R(FRMSZ+I-M+1) (MINUS SPACING) IS ALREADY IN BR1)
                     (04558) ;
  A17 06BFE 2E600000 (04559) APCS$U: LOAD(BR2,M$$)               BR2 <= ADDR U(0)
  A18 06C00 30220002 (04560)          SUB(BR2,W$)                    BR2 <= ADDR U(0) - SPACING
                     (04561) ;
                     (04562) ;
                     (04563) ; START LOOP
                     (04564) ;  GEN U(I),   T(0),V(14),...,T(5),V(9)
                     (04565) ;
  A19 06C02 32AA0002 (04566) @1       ADD(BR2,W$,TF)                 GEN ADDR U(I)
                     (04567) ;
                     (04568) APCS$IT0:
  A1A 06C04 34C00000 (04569)          LOAD(BR0,M$$,TF)               GEN ADDR T(0)
                     (04570) APCS$V14:
  A1B 06C06 36C00000 (04571)          LOAD(BR0,M$$,TF)               GEN ADDR V(14)
                     (04572) APCS$IT1:
  A1C 06C08 38C00000 (04573)          LOAD(BR0,M$$,TF)               GEN ADDR T(1)
                     (04574) APCS$V13:
  A1D 06C0A 3AC00000 (04575)          LOAD(BR0,M$$,TF)               GEN ADDR V(13)
                     (04576) APCS$IT2:
  A1E 06C0C 3CC00000 (04577)          LOAD(BR0,M$$,TF)               GEN ADDR T(2)
                     (04578) APCS$V12:
  A1F 06C0E 3EC00000 (04579)          LOAD(BR0,M$$,TF)               GEN ADDR V(12)
```

```
                      (04580) APCS$IT3:
A20 06C10 40C00000 (04581)          LOAD(BR0,M$$,TF)                GEN ADDR T(3)
                      (04582) APCS$V11:
A21 06C12 42C00000 (04583)          LOAD(BR0,M$$,TF)                GEN ADDR V(11)
                      (04584) APCS$IT4:
A22 06C14 44C00000 (04585)          LOAD(BR0,M$$,TF)                GEN ADDR T(4)
                      (04586) APCS$V10:
A23 06C16 46C00000 (04587)          LOAD(BR0,M$$,TF)                GEN ADDR V(10)
                      (04588) APCS$IT5:
A24 06C18 48C00000 (04589)          LOAD(BR0,M$$,TF)                GEN ADDR T(5)
                      (04590) APCS$V9:
A25 06C1A 4AC00000 (04591)          LOAD(BR0,M$$,TF)                GEN ADDR V(9)
                      (04592) ;
                      (04593) ; GEN V(5),...,V(0)
                      (04594) ;
                      (04595)'APCS$V5:
A26 06C1C 4CC00000 (04596)          LOAD(BR0,M$$,TF)                GEN ADDR V(5)
A27 06C1E 4E020002 (04597)          SUB(BR0,W$,TF)                  GEN ADDR V(4)
A28 06C20 50020002 (04598)          SUB(BR0,W$,TF)                  GEN ADDR V(3)
A29 06C22 52020002 (04599)          SUB(BR0,W$,TF)                  GEN ADDR V(2)
A2A 06C24 54020002 (04600)          SUB(BR0,W$,TF)                  GEN ADDR V(1)
A2B 06C26 56020002 (04601)          SUB(BR0,W$,TF)                  GEN ADDR V(0)
                      (04602) ;
                      (04603) ; GEN V(6),V(7),V(8)
                      (04604) ;
                      (04605) APCS$V6:
A2C 06C28 58C00000 (04606)          LOAD(BR0,M$$,TF)                GEN ADDR V(6)
A2D 06C2A 5A0A0002 (04607)          ADD(BR0,W$,TF)                  GEN ADDR V(7)
A2E 06C2C 5C0A0002 (04608)          ADD(BR0,W$,TF)                  GEN ADDR V(8)
                      (04609) ;
                      (04610) ; GEN R(FRMSZ+I-M+1)
                      (04611) ;
A2F 06C2E 5E9A0002 (04612)          ADD(FR1,W$,TF)                  GEN ADDR R(FRMSZ+I-M+1)
                      (04613) ;
                      (04614) ; FOR   I=0:71 GEN: W(1),CODING THRESH,(2**-15),W(0)
                      (04615) ;       I=72:143  : W(3),CODING THRESH,(2**-15),W(2)
                      (04616) ;       I=144:215 : W(5),CODING THRESH,(2**-15),W(4)
                      (04617) ;
                      (04618) ; I=0:71    => ITERATION COUNTER = BR3 = (FRMSZ-1):(FRMSZ-72) = (215:144)
                      (04619) ; I=72:143 => ITERATION COUNTER = BR3 = (FRMSZ-73):(FRMSZ-144) = (143:72)
                      (04620) ; I=144:215=> ITERATION COUNTER = BR3 = (FRMSZ-145):(FRMSZ-216) = (71:0)
                      (04621) ;
                      (04622) ; SEE IF BR3 IS IN (215:144) => W(1)/W(0)
                      (04623) ;
A30 06C30 6040008F (04624)          LOAD(BR0,143)
A31 06C32 620937A7 (04625)          SUBB(BR0,BR3),JUMPP(APCS$2)     BR0 <= 143 - (ITER CTR)
                      (04626) ;                                       JUMP IF BR3 NOT IN (215:144)
                      (04627) APCS$W1:
A32 06C34 64C00000 (04628)          LOAD(BR0,M$$,TF)                GEN W(1)
A33 06C36 66C299FC (04629)          LOAD(BR0,CTHU+W$+W$(1),TF)      GEN POSITIVE VALUED CODING THRESH
A34 06C38 68C203CE (04630)          LOAD(BR0,SVT$UN1(1),TF)         GEN (2**-15)
                      (04631) APCS$W0:
A35 06C3A 6AC00000 (04632)          LOAD(BR0,M$$,TF)                GEN V(0)
```

```
PAGE  104:  [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APS3 - APC(Y,A,U,V,W,R,S,T)   APC (WITH NOISE SHAPING)

 A36 06C3C 6C004360 (04633)         JUMP(APCS$4)                   JUMP TO COMMON CODE
                    (04634) ;
                    (04635) ; HERE IF NOT W(1)/W(0)
                    (04636) ; SEE IF BR3 IS IN (143:72) => W(3)/W(2)
                    (04637) ;
 A37 06C3E 6E400047 (04638) APCS$2: LOAD(BR0,71)
 A38 06C40 70093EA7 (04639)         SUBB(BR0,BR3),JUMPP(APCS$3)    BR0 <= 71 - (ITER CTR)
                    (04640) ;                                      JUMP IF BR3 NOT IN (143:72)
                    (04641) APCS$W3:
 A39 06C42 72C00000 (04642)         LOAD(BR0,M$$,TF)               GEN W(3)
 A3A 06C44 74C299FC (04643)         LOAD(BR0,CTHW+W$+W$(1),TF)     GEN POSITIVE VALUED CODING THRESH
 A3B 06C46 76C203CE (04644)         LOAD(BR0,SVT$UN1(1),TF)        GEN (2**-15)
                    (04645) APCS$W2:
 A3C 06C48 78C00000 (04646)         LOAD(BR0,M$$,TF)               GEN W(2)
 A3D 06C4A 7A004360 (04647)         JUMP(APCS$4)                   JUMP TO COMMON CODE
                    (04648) ;
                    (04649) ; HERE IF NOT W(1)/W(0) OR W(3)/W(2)
                    (04650) ; MUST THEREFORE BE W(5)/W(4)
                    (04651) ;
                    (04652) APCS$3:
                    (04653) APCS$W5:
 A3E 06C4C 7CC00000 (04654)         LOAD(BR0,M$$,TF)               GEN W(5)
 A3F 06C4E 7EC299FC (04655)         LOAD(BR0,CTHW+W$+W$(1),TF)     GEN POSITIVE VALUED CODING THRESH
 A40 06C50 80C203CE (04656)         LOAD(BR0,SVT$UN1(1),TF)        GEN (2**-15)
                    (04657) APCS$W4:
 A41 06C52 82C00000 (04658)         LOAD(BR0,M$$,TF)               GEN W(4)
 A42 06C54 84004360 (04659)         JUMP(APCS$4)                   JUMP TO COMMON CODE
                    (04660) ;
                    (04661) ;
                    (04662) ; COMMON CODE:
                    (04663) ;  GEN QUANT VALS (1-4)
                    (04664) ;
 A43 06C56 86C29BF0 (04665) APCS$4: LOAD(BR0,DVLU(1),TF)           GEN QUANT VAL(1)
 A44 06C58 888A0002 (04666)         ADD(BR0,W$,TF)                 GEN QUANT VAL(2)
 A45 06C5A 8A8A0002 (04667)         ADD(BR0,W$,TF)                 GEN QUANT VAL(3)
 A46 06C5C 8C8A0002 (04668)         ADD(BR0,W$,TF)                 GEN QUANT VAL(4)
                    (04669) ;
                    (04670) ; END OF LOOP:
                    (04671) ;  DECREMENT ITERATION COUNTER (BR3).
                    (04672) ;  IF RESULT IS NEGATIVE, DONE.
                    (04673) ;
 A47 06C5E 8E391901 (04674)         SUBL(BR3,1),JUMPP(#1)
                    (04675) ;
 A48 06C60 90200031 (04676)         CLEAR(RI)
 A49 06C62 92000020 (04677)         NOP
                    (04678)         EJECT
```

```
                    (04679) ;
                    (04680) ;
                    (04681) ;
                    (04682) ; OUTPUT PROGRAM
                    (04683) ;       BW0: SCRATCH
                    (04684) ;       BW1: R(FRMSZ+I)
                    (04685) ;       BW2: Y(I)
                    (04686) ;       BW3: OUTPUT LOOP ITERATION COUNTER ((FRMSZ-1):0)
                    (04687) ;
A4A 06C64 94300032  (04688) APCS$O: SET(RA)
                    (04689) ;
                    (04690) ;
                    (04691) ; GEN BUS1 ADDR FOR INSTRUCTION TO BE MODIFIED (RIGHT HALF)
                    (04692) ;
A4B 06C66 97726BE7  (04693)         LOAD(BW3,APCS$MOD+W$(1),TE)
                    (04694) ;
                    (04695) ; GEN PSEUDO-MEM ADDR FOR LOADING APS MEMORY
                    (04696) ;
A4C 06C68 9BF3FFC8  (04697)         LOAD(BW3,APSMEM(1),TF)
                    (04698) ;
                    (04699) ;
                    (04700) ; INITIALIZE OUTPUT LOOP
                    (04701) ;  KEEP ITERATION COUNTER IN BW3 ((FRMSZ-1):0)
                    (04702) ;
A4D 06C6A 9A7000D7  (04703)         LOAD(BW3,FRMSZ-1)               BW3 <= FRMSZ-1
                    (04704) ;
                    (04705) ; GET INITIAL Y BUFFER ADDRESS (MINUS SPACING)
                    (04706) ;
A4E 06C6C 9C600000  (04707) APCS$Y: LOAD(BW2,M$$)               BW2 <= ADDR Y(0)
A4F 06C6E 9E220001  (04708)         SUB(BW2,M$)                     BW2 <= ADDR Y(0) - SPACING
                    (04709) ;
                    (04710) ; GET INITIAL R(FRMSZ+I)  (MINUS SPACING)
                    (04711) ;
                    (04712) APCS$UR:
A50 06C70 A0500000  (04713)         LOAD(BW1,M$$)                   BW1 <= ADDR R(0)
A51 06C72 A2120002  (04714)         SUB(BW1,W$)                             - SPACING
A52 06C74 A41A01B0  (04715)         ADD(BW1,FRMSZ*W$)               BW1 <= ADDR R(FRMSZ+0) - SPACING
                    (04716) ;
                    (04717) ;
                    (04718) ; START LOOP
                    (04719) ;  GEN T(0),...,T(4)
                    (04720) ;
                    (04721) B2
                    (04722) APCS$OT0:
A53 06C76 A6C00000  (04723)         LOAD(BW0,M$$,TF)                GEN ADDR T(0)
A54 06C78 A80A0002  (04724)         ADD(BW0,W$,TF)                  GEN ADDR T(1)
A55 06C7A AA0A0002  (04725)         ADD(BW0,W$,TF)                  GEN ADDR T(2)
A56 06C7C AC0A0002  (04726)         ADD(BW0,W$,TF)                  GEN ADDR T(3)
A57 06C7E AE0A0002  (04727)         ADD(BW0,W$,TF)                  GEN ADDR T(4)
                    (04728) ;
                    (04729) ; GEN Y(I)
                    (04730) ;
A58 06C80 B12A0001  (04731)         ADD(BW2,M$,TE)                  GEN ADDR Y(I)
```

```
PAGE  106:  [BBND]<DCA16>BBN16N.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
             APS3 - APC(Y,A,U,V,W,R,S,T)   APC (WITH NOISE SHAPING)

                      (04732) ;
                      (04733) ; GEN T(5)
                      (04734) ;
 A59 06C82 B20A0002 (04735)         ADD(BW0,W$,TF)                 GEN ADDR T(5)
                      (04736) ;
                      (04737) ; GEN R(FRMSZ+I)
                      (04738) ;
 A5A 06C84 B49A0002 (04739)         ADD(BW1,W$,TF)                 GEN ADDR R(FRMSZ+I)
                      (04740) ;
                      (04741) ; END OF LOOP:
                      (04742) ;  DECREMENT ITERATION COUNTER (BW3)
                      (04743) ;  IF RESULT IS NEGATIVE, DONE.
                      (04744) ;
 A5B 06C86 86315381 (04745)         SUBL(BW3,1),JUMPP(#2)
                      (04746) ;
                      (04747) ; AFTER LOOP, CLEAN UP:
                      (04748) ;  GEN S(0),...,S(5)
                      (04749) ;
                      (04750) APCS$OS:
 A5C 06C88 B0C00000 (04751)         LOAD(BW0,M$$,TF)               GEN ADDR S(0)
 A5D 06C8A BA8A0002 (04752)         ADD(BW0,W$,TF)                 GEN ADDR S(1)
 A5E 06C8C BC8A0002 (04753)         ADD(BW0,W$,TF)                 GEN ADDR S(2)
 A5F 06C8E BE8A0002 (04754)         ADD(BW0,W$,TF)                 GEN ADDR S(3)
 A60 06C90 C08A0002 (04755)         ADD(BW0,W$,TF)                 GEN ADDR S(4)
 A61 06C92 C28A0002 (04756)         ADD(BW0,W$,TF)                 GEN ADDR S(5)
                      (04757) ;
                      (04758) ; DONE.
                      (04759) ;
 A62 06C94 C4200030 (04760)         CLEAR(RO)        ;DONE WITH OUTPUT
 A63 06C96 C6000020 (04761)         NOP(0)
           00000000 (04762) APCS$C=#C
     06C98          (04763)         END
                      (04764) ;
           000000C8 (04765) APCS$Z=#L-APCS$
```

```
                   (04766) * SBM$APC  SPECIAL BINDING MODULE FOR APC
                   (04767) ; KFIELD 8/80
                   (04768) ;
                   (04769) ;       FCB FORMAT (16 BIT FORMAT SHOWN)
                   (04770) ;
                   (04771) ;     WORD      LEFT BYTE          RIGHT BYTE
                   (04772) ;     ----      ---------          ----------
                   (04773) ;      0:        PTR TO NEXT FCB AND FUNCTION LIST FLAG (LSB)
                   (04774) ;      1:          199                CONTROL BITS
                   (04775) ;      2:           Y                   A
                   (04776) ;      3:           U                   V
                   (04777) ;      4:           W                   R
                   (04778) ;      5:           S                   T
                   (04779) ;
                   (04780) ;!!! ALL BUFFERS MUST BE RL,LNG,CONTIGUOUS,
                   (04781) ;!!! EXCEPT BUFFER Y WHICH MUST BE FXD,LNG,CNTG.
                   (04782) ;
                   (04783) SBM$APC:
06C98 C63400E8 (04784)          PUSHMIL R3,APDT$ORG(R2)      STACK APU MODULE BUS ORIGIN
06C9A 9071FFFE (04785)          MOVIR R7,-W$
06C9C C6D7FFFE (04786)          PUSHMIL R3,@-W$(R3)          STACK APU MODULE ST ADDR AND SZ
06C9E C63400EA (04787)          PUSHMIL R3,APDT$ORG+W$(R2)   STACK APS MODULE BUS ORIGIN
06CA0 C63400EC (04788)          PUSHMIL R3,APDT$ORG+2*W$(R2) STACK CSPU SUPPORT MODULE
06CA2 2771     (04789)          DECR R7,1
06CA3 0000     (04790)          EVEN
06CA4 C4D7FFFC (04791)          PUSHMI R3,@-2*W$(R3)         APS MODULE SIZE
06CA6 9047FFFD (04792)          MOVIR R4,-3*R$(R3)           GET SPECIAL SUPPORT SEMAPHORE
06CA8 7040FF00 (04793)          MOVWK R4,R4,MSK$LBYT
06CAA 3C48     (04794)          LRS R4,8
06CAB 3430     (04795)          PUSHXI R3,R4                 STACK SPECIAL SUPPORT SEMAPHORE
06CAC 4052     (04796)          MOVRR R5,R1                  POINT TO FIRST SCALAR ID
06CAD 2651     (04797)          INCR R5,H$
06CAE 704A00FF (04798)          MOVWK R4,R5,MSK$RBYT         EXTRACT SCALAR A IDENTIFIER
06CB0 3630     (04799)          PUSHXIL R3,R4                STACK SCALAR A IDENTIFIER
06CB1 C6300792 (04800)          PUSHMIL R3,CLR$GBG1          RESET G FLAGS
06CB3 2632     (04801)          INCR R3,2
06CB4 C63007BA (04802)          PUSHMIL R3,ZERO              SET FOR NO PBF
06CB6 9E300011 (04803)          SUBIR R3,APS$LLOC
                   (04804) ;
                   (04805) ;
                   (04806) ; SCALAR A BINDING
                   (04807) ;
06CB8 F8420001 (04808)         MOVMR    R4,H$(R1)               SCALAR A ID
06CBA 9A4000FF (04809)         ANDIR    R4,MSK$RBYT
06CBC 3A41     (04810)         LLS      R4,1                    *2 FOR FWORD SCALAR TABLE
06CBD 9C400382 (04811)         ADDIR    R4,SVT$                 ADD TO TOP OF TABLE
06CBF E0406807 (04812)         MOVRM    R4,APCS$+W$*APCS$SA+H$  ;INSERT INTO APS PROGRAM
                   (04813) ;
                   (04814) ; Y BASE BINDING  !!SPECIAL FIX NEEDED FOR LNG FXD BUFFER!!
                   (04815) ;
06CC1 F8620001 (04816)         MOVMR    R6,H$(R1)
06CC3 596CFF00 (04817)         MOVKR    R6,R6,MSK$LBYT          ;GET BID
06CC5 3C67     (04818)         LRS      R6,7                    ;CONVERT TO BCT INDEX
```

```
                    (04819)         EVEN
06CC6 C04C0582 (04820)         MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06CC8 F0706C6C (04821)         MOVMR   R7,APCS$+M$*APCS$Y       ;GET LEFT HWORD OF INSTR
06CCA 564FFFF0 (04822)         IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
06CCC 2430     (04823)         CRB     BIT$MS,R4                ;ASSUME FID,LNG: CLR ML BIT
06CCD 04406C6C (04824)         MOVRML  R4,APCS$+M$*APCS$Y       ;MOVE FWORD TO BUS1
               (04825) ;
               (04826) ; W BASE BINDING
               (04827) ;
06CCF F0620002 (04828)         MOVMR   R6,2*M$(R1)
06CD1 5060FF00 (04829)         MOVKR   R6,R6,MSK$LBYT           ;GET BID
06CD3 3C67     (04830)         LRS     R6,7                     ;CONVERT TO BCT INDEX
               (04831)         EVEN
06CD4 C04C0582 (04832)         MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06CD6 F0706BFE (04833)         MOVMR   R7,APCS$+M$*APCS$W       ;GET LEFT HWORD OF INSTR
06CD8 564FFFF0 (04834)         IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
06CDA 04406BFE (04835)         MOVRML  R4,APCS$+M$*APCS$W       ;MOVE FWORD TO BUS1
               (04836) ;
               (04837) ; V BASE BINDING
               (04838) ;
06CDC F0620002 (04839)         MOVMR   R6,2*M$(R1)
06CDE 506C00FF (04840)         MOVKR   R6,R6,MSK$RBYT           ;GET BID
06CE0 3A61     (04841)         LLS     R6,1                     ;CONVERT TO BCT INDEX
06CE1 0000     (04842)         EVEN
               (04843) ; V(5)... (TEMPORARILY PUT V(0) ADDR WHERE V(5) WILL GO)
06CE2 C04C0582 (04844)         MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06CE4 F0706C1C (04845)         MOVMR   R7,APCS$+M$*APCS$V5      ;GET LEFT HWORD OF INSTR
06CF6 564EFFF0 (04846)         IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
06CF8 04406C1C (04847)         MOVRML  R4,APCS$+M$*APCS$V5      ;MOVE FWORD TO BUS1
06CEA EE706C1C (04848)         LAF     R7,APCS$+M$*APCS$V5      ;GET ADDR OF BUFFER START
06CEC 9C70000A (04849)         ADDIR   R7,5*M$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06CEE EF706C1C (04850)         SAF     R7,APCS$+M$*APCS$V5      ;STORE ADDR OF DESIRED SAMPLE
               (04851) ;
               (04852) ; V(6)...
06CF0 C04C0582 (04853)         MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06CF2 F0706C20 (04854)         MOVMR   R7,APCS$+M$*APCS$V6      ;GET LEFT HWORD OF INSTR
06CF4 564EFFF0 (04855)         IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
06CF6 04406C20 (04856)         MOVRML  R4,APCS$+M$*APCS$V6      ;MOVE FWORD TO BUS1
06CF8 EE706C20 (04857)         LAF     R7,APCS$+M$*APCS$V6      ;GET ADDR OF BUFFER START
06CFA 9C70000C (04858)         ADDIR   R7,6*M$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06CFC EF706C20 (04859)         SAF     R7,APCS$+M$*APCS$V6      ;STORE ADDR OF DESIRED SAMPLE
               (04860) ;
               (04861) ; V(9)...
06CFE C04C0582 (04862)         MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06D00 F0706C1A (04863)         MOVMR   R7,APCS$+M$*APCS$V9      ;GET LEFT HWORD OF INSTR
06D02 564FFFF0 (04864)         IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
06D04 04406C1A (04865)         MOVRML  R4,APCS$+M$*APCS$V9      ;MOVE FWORD TO BUS1
06D06 EE706C1A (04866)         LAF     R7,APCS$+M$*APCS$V9      ;GET ADDR OF BUFFER START
06D08 9C700012 (04867)         ADDIR   R7,9*M$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06D0A EF706C1A (04868)         SAF     R7,APCS$+M$*APCS$V9      ;STORE ADDR OF DESIRED SAMPLE
               (04869) ;
               (04870) ; V(10)...
06D0C C04C0582 (04871)         MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
```

```
006D0E F0706C16 (04872)         MOVMR    R7,APCS$+W$*APCS$V10      ;GET LEFT HWORD OF INSTR
006D10 564FFFF0 (04873)         IORKR    R4,R7,$FFF0               ;'OR' IN HIGH 4 BITS
006D12 04406C16 (04874)         MOVRML   R4,APCS$+W$*APCS$V10      ;MOVE FWORD TO BUS1
006D14 EF706C16 (04875)         LAF      R7,APCS$+W$*APCS$V10      ;GET ADDR OF BUFFER START
006D16 9C700014 (04876)         ADDIR    R7,10*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
006D18 EF706C16 (04877)         SAF      R7,APCS$+W$*APCS$V10      ;STORE ADDR OF DESIRED SAMPLE
                (04878) ;
                (04879) ; V(11)...
006D1A C04C0582 (04880)         MOVMRL   R4,BCT$BA(R6)             ;1BIT LEN., 2BIT BUS#, 17BIT BA.
006D1C F0706C12 (04881)         MOVMR    R7,APCS$+W$*APCS$V11      ;GET LEFT HWORD OF INSTR
006D1E 564FFFF0 (04882)         IORKR    R4,R7,$FFF0               ;'OR' IN HIGH 4 BITS
006D20 04406C12 (04883)         MOVRML   R4,APCS$+W$*APCS$V11      ;MOVF FWORD TO BUS1
006D22 EE706C12 (04884)         LAF      R7,APCS$+W$*APCS$V11      ;GET ADDR OF BUFFER START
006D24 9C700016 (04885)         ADDIR    R7,11*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
006D26 EF706C12 (04886)         SAF      R7,APCS$+W$*APCS$V11      ;STORE ADDR OF DESIRED SAMPLE
                (04887) ;
                (04888) ; V(12)...
006D28 C04C0582 (04889)         MOVMRL   R4,BCT$BA(R6)             ;1BIT LEN., 2BIT BUS#, 17BIT BA.
006D2A F0706C0E (04890)         MOVMR    R7,APCS$+W$*APCS$V12      ;GET LEFT HWORD OF INSTR
006D2C 564FFFF0 (04891)         IORKR    R4,R7,$FFF0               ;'OR' IN HIGH 4 BITS
006D2E 04406C0E (04892)         MOVRML   R4,APCS$+W$*APCS$V12      ;MOVE FWORD TO BUS1
006D30 EF706C0E (04893)         LAF      R7,APCS$+W$*APCS$V12      ;GET ADDR OF BUFFER START
006D32 9C700018 (04894)         ADDIR    R7,12*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
006D34 EF706C0E (04895)         SAF      R7,APCS$+W$*APCS$V12      ;STORE ADDR OF DESIRED SAMPLE
                (04896) ;
                (04897) ; V(13)...
006D36 C04C0582 (04898)         MOVMRL   R4,BCT$BA(R6)             ;1BIT LEN., 2BIT BUS#, 17BIT BA.
006D38 F0706C0A (04899)         MOVMR    R7,APCS$+W$*APCS$V13      ;GET LEFT HWORD OF INSTR
006D3A 564FFFF0 (04900)         IORKR    R4,R7,$FFF0               ;'OR' IN HIGH 4 BITS
006D3C 04406C0A (04901)         MOVRML   R4,APCS$+W$*APCS$V13      ;MOVE FWORD TO BUS1
006D3E EE706C0A (04902)         LAF      R7,APCS$+W$*APCS$V13      ;GET ADDR OF BUFFER START
006D40 9C70001A (04903)         ADDIR    R7,13*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
006D42 EF706C0A (04904)         SAF      R7,APCS$+W$*APCS$V13      ;STORE ADDR OF DESIRED SAMPLE
                (04905) ;
                (04906) ; V(14)...
006D44 C04C0582 (04907)         MOVMRL   R4,BCT$BA(R6)             ;1BIT LEN., 2BIT BUS#, 17BIT BA.
006D46 F0706C06 (04908)         MOVMR    R7,APCS$+W$*APCS$V14      ;GET LEFT HWORD OF INSTR
006D48 564FFFF0 (04909)         IORKR    R4,R7,$FFF0               ;'OR' IN HIGH 4 BITS
006D4A 04406C06 (04910)         MOVRML   R4,APCS$+W$*APCS$V14      ;MOVE FWORD TO BUS1
006D4C EE706C06 (04911)         LAF      R7,APCS$+W$*APCS$V14      ;GET ADDR OF BUFFER START
006D4E 9C70001C (04912)         ADDIR    R7,14*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
006D50 EF706C06 (04913)         SAF      R7,APCS$+W$*APCS$V14      ;STORE ADDR OF DESIRED SAMPLE
                (04914) ;
                (04915) ; W BASE BINDING
                (04916) ;
006D52 F0620003 (04917)         MOVMR    R6,3*H$(R1)
006D54 506CFF00 (04918)         MOVKR    R6,R6,MSK$LBYT            ;GET BID
006D56 3C67     (04919)         LRS      R6,7                      ;CONVERT TO BCT INDEX
006D57 0000     (04920)         EVEN
                (04921) ;
                (04922) ; W(0)...
006D58 C04C0582 (04923)         MOVMRL   R4,BCT$BA(R6)             ;1BIT LEN., 2BIT BUS#, 17BIT BA.
006D5A F0706C3A (04924)         MOVMR    R7,APCS$+W$*APCS$W0       ;GET LEFT HWORD OF INSTR
```

```
06D5C 564EFFF0 (04925)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
06D5E 84406C3A (04926)          MOVRML  R4,APCS$+W$*APCS$W0      ;MOVE FWORD TO BUS1
               (04927) ;
               (04928) ; W(1)...
06D60 C04C0582 (04929)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06D62 F0706C34 (04930)          MOVMR   R7,APCS$+W$*APCS$W1      ;GET LEFT HWORD OF INSTR
06D64 564EFFF0 (04931)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
06D66 84406C34 (04932)          MOVRML  R4,APCS$+W$*APCS$W1      ;MOVE FWORD TO BUS1
06D68 EF706C34 (04933)          LAF     R7,APCS$+W$*APCS$W1      ;GET ADDR OF BUFFER START
06D6A 9C700002 (04934)          ADDIR   R7,1*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06D6C EF706C34 (04935)          SAF     R7,APCS$+W$*APCS$W1      ;STORE ADDR OF DESIRED SAMPLE
               (04936) ;
               (04937) ; W(2)...
06D6E C04C0582 (04938)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06D70 F0706C40 (04939)          MOVMR   R7,APCS$+W$*APCS$W2      ;GET LEFT HWORD OF INSTR
06D72 564EFFF0 (04940)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
06D74 84406C40 (04941)          MOVRML  R4,APCS$+W$*APCS$W2      ;MOVE FWORD TO BUS1
06D76 EE706C40 (04942)          LAF     R7,APCS$+W$*APCS$W2      ;GET ADDR OF BUFFER START
06D78 9C700004 (04943)          ADDIR   R7,2*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06D7A EF706C40 (04944)          SAF     R7,APCS$+W$*APCS$W2      ;STORE ADDR OF DESIRED SAMPLE
               (04945) ;
               (04946) ; W(3)...
06D7C C04C0582 (04947)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06D7E F0706C42 (04948)          MOVMR   R7,APCS$+W$*APCS$W3      ;GET LEFT HWORD OF INSTR
06D80 564EFFF0 (04949)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
06D82 84406C42 (04950)          MOVRML  R4,APCS$+W$*APCS$W3      ;MOVE FWORD TO BUS1
06D84 EF706C42 (04951)          LAF     R7,APCS$+W$*APCS$W3      ;GET ADDR OF BUFFER START
06D86 9C700006 (04952)          ADDIR   R7,3*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06D88 EF706C42 (04953)          SAF     R7,APCS$+W$*APCS$W3      ;STORE ADDR OF DESIRED SAMPLE
               (04954) ;
               (04955) ; W(4)...
06D8A C04C0582 (04956)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06D8C F0706C52 (04957)          MOVMR   R7,APCS$+W$*APCS$W4      ;GET LEFT HWORD OF INSTR
06D8E 564EFFF0 (04958)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
06D90 84406C52 (04959)          MOVRML  R4,APCS$+W$*APCS$W4      ;MOVE FWORD TO BUS1
06D92 EE706C52 (04960)          LAF     R7,APCS$+W$*APCS$W4      ;GET ADDR OF BUFFER START
06D94 9C700008 (04961)          ADDIR   R7,4*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06D96 EF706C52 (04962)          SAF     R7,APCS$+W$*APCS$W4      ;STORE ADDR OF DESIRED SAMPLE
               (04963) ;
               (04964) ; W(5)...
06D98 C04C0582 (04965)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06D9A F0706C4C (04966)          MOVMR   R7,APCS$+W$*APCS$W5      ;GET LEFT HWORD OF INSTR
06D9C 564EFFF0 (04967)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
06D9E 84406C4C (04968)          MOVRML  R4,APCS$+W$*APCS$W5      ;MOVE FWORD TO BUS1
06DA0 EE706C4C (04969)          LAF     R7,APCS$+W$*APCS$W5      ;GET ADDR OF BUFFER START
06DA2 9C70000A (04970)          ADDIR   R7,5*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
06DA4 EF706C4C (04971)          SAF     R7,APCS$+W$*APCS$W5      ;STORE ADDR OF DESIRED SAMPLE
               (04972) ;
               (04973) ; R BASE BINDING
               (04974) ;
06DA6 F0620003 (04975)          MOVMR   R6,3*HS(R1)
06DA8 506C00FF (04976)          MOVKR   R6,R6,MSK$RBYT           ;GET BID
06DAA 3A61     (04977)          LLS     R6,1                     ;CONVERT TO BCT INDEX
```

```
PAGE  111:  (BBND)<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            SOM$APC  SPECIAL BINDING MODULE FOR APC

      06DAD 0000     (04978)          EVEN
                     (04979) ;
                     (04980) ; R(0) (INPUT)...
      06DAC C04C0582 (04981)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
      06DAE F0706BE2 (04982)          MOVMR   R7,APCS$+W$*APCS$IR      ;GET LEFT HWORD OF INSTR
      06DB0 564EFFF0 (04983)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
      06DB2 04406BE2 (04984)          MOVRML  R4,APCS$+W$*APCS$IR      ;MOVE FWORD TO BUS1
                     (04985) ;
                     (04986) ; R(0) (OUTPUT)...
      06DB4 C04C0582 (04987)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
      06DB6 F0706C70 (04988)          MOVMR   R7,APCS$+W$*APCS$OR      ;GET LEFT HWORD OF INSTR
      06DB8 564EFFF0 (04989)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
      06DBA 04406C70 (04990)          MOVRML  R4,APCS$+W$*APCS$OR      ;MOVE FWORD TO BUS1
                     (04991) ;
                     (04992) ; S BASE BINDING
                     (04993) ;
      06DBC F0620004 (04994)          MOVMR   R6,4*H$(R1)
      06DBE 506CFF00 (04995)          MOVKR   R6,R6,MSK$LBYT           ;GET BID
      06DC0 3C67     (04996)          LRS     R6,7                     ;CONVERT TO BCT INDEX
      06DC1 0000     (04997)          EVEN
                     (04998) ;
                     (04999) ; S(0) (INPUT)...
      06DC2 C04C0582 (05000)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
      06DC4 F0706BF0 (05001)          MOVMR   R7,APCS$+W$*APCS$IS      ;GET LEFT HWORD OF INSTR
      06DC6 564EFFF0 (05002)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
      06DC8 04406BF0 (05003)          MOVRML  R4,APCS$+W$*APCS$IS      ;MOVE FWORD TO BUS1
                     (05004) ;
                     (05005) ; S(0) (OUTPUT)...
      06DCA C04C0582 (05006)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
      06DCC F0706C80 (05007)          MOVMR   R7,APCS$+W$*APCS$OS      ;GET LEFT HWORD OF INSTR
      06DCE 564EFFF0 (05008)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
      06DD0 04406C80 (05009)          MOVRML  R4,APCS$+W$*APCS$OS      ;MOVE FWORD TO BUS1
                     (05010) ;
                     (05011) ; T BASE BINDING
                     (05012) ;
      06DD2 F0620004 (05013)          MOVMR   R6,4*H$(R1)
      06DD4 506C00FF (05014)          MOVKR   R6,R6,MSK$RBYT           ;GET BID
      06DD6 3A61     (05015)          LLS     R6,1                     ;CONVERT TO BCT INDEX
      06DD7 000F     (05016)          EVEN
                     (05017) ;
                     (05018) ; T(0) (INPUT)...
      06DD8 C04C0582 (05019)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
      06DDA F0706C04 (05020)          MOVMR   R7,APCS$+W$*APCS$IT0     ;GET LEFT HWORD OF INSTR
      06DDC 564FFFF0 (05021)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
      06DDE 04406C04 (05022)          MOVRML  R4,APCS$+W$*APCS$IT0     ;MOVE FWORD TO BUS1
                     (05023) ;
                     (05024) ; T(1) (INPUT)...
      06DE0 C04C0582 (05025)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
      06DE2 F0706C08 (05026)          MOVMR   R7,APCS$+W$*APCS$IT1     ;GET LEFT HWORD OF INSTR
      06DE4 564EFFF0 (05027)          IORKR   R4,R7,$FFF0              ;'OR' IN HIGH 4 BITS
      06DE6 04406C08 (05028)          MOVRML  R4,APCS$+W$*APCS$IT1     ;MOVE FWORD TO BUS1
      06DE8 EF706C08 (05029)          LAF     R7,APCS$+W$*APCS$IT1     ;GET ADDR OF BUFFER START
      06DEA 9C700002 (05030)          ADDIR   R7,1*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
```

```
PAGE  112: [BBND]<DCA16>BBN16M.MSU.2, 29-Dec-80 14:30:40, Ed: WOLF
           SBM$APC  SPECIAL BINDING MODULE FOR APC

     06DFC EF706C08 (05031)          SAF     R7,APCS$+W$*APCS$IT1     ;STORE ADDR OF DESIRED SAMPLE
                    (05032) ;
                    (05033) ; T(2) (INPUT)...
     06DEE C04C0582 (05034)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
     06DF0 F0706C0C (05035)          MOVMR   R7,APCS$+W$*APCS$IT2     ;GET LEFT HWORD OF INSTR
     06DF2 564EFFF0 (05036)          IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
     06DF4 84406C0C (05037)          MOVRML  R4,APCS$+W$*APCS$IT2     ;MOVE FWORD TO BUS1
     06DF6 EE706C0C (05038)          LAF     R7,APCS$+W$*APCS$IT2     ;GET ADDR OF BUFFER START
     06DF8 9C700004 (05039)          ADDIR   R7,2*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
     06DFA EF706C0C (05040)          SAF     R7,APCS$+W$*APCS$IT2     ;STORE ADDR OF DESIRED SAMPLE
                    (05041) ;
                    (05042) ; T(3) (INPUT)...
     06DFC C04C0582 (05043)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
     06DFE F0706C10 (05044)          MOVMR   R7,APCS$+W$*APCS$IT3     ;GET LEFT HWORD OF INSTR
     06E00 564EFFF0 (05045)          IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
     06E02 84406C10 (05046)          MOVRML  R4,APCS$+W$*APCS$IT3     ;MOVE FWORD TO BUS1
     06E04 EE706C10 (05047)          LAF     R7,APCS$+W$*APCS$IT3     ;GET ADDR OF BUFFER START
     06E06 9C700006 (05048)          ADDIR   R7,3*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
     06E08 EF706C10 (05049)          SAF     R7,APCS$+W$*APCS$IT3     ;STORE ADDR OF DESIRED SAMPLE
                    (05050) ;
                    (05051) ; T(4) (INPUT)...
     06E0A C04C0582 (05052)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
     06E0C F0706C14 (05053)          MOVMR   R7,APCS$+W$*APCS$IT4     ;GET LEFT HWORD OF INSTR
     06E0E 564EFFF0 (05054)          IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
     06E10 84406C14 (05055)          MOVRML  R4,APCS$+W$*APCS$IT4     ;MOVE FWORD TO BUS1
     06E12 EE706C14 (05056)          LAF     R7,APCS$+W$*APCS$IT4     ;GET ADDR OF BUFFER START
     06E14 9C700008 (05057)          ADDIR   R7,4*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
     06E16 EF706C14 (05058)          SAF     R7,APCS$+W$*APCS$IT4     ;STORE ADDR OF DESIRED SAMPLE
                    (05059) ;
                    (05060) ; T(5) (INPUT)...
     06E18 C04C0582 (05061)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
     06E1A F0706C18 (05062)          MOVMR   R7,APCS$+W$*APCS$IT5     ;GET LEFT HWORD OF INSTR
     06E1C 564EFFF0 (05063)          IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
     06E1E 84406C18 (05064)          MOVRML  R4,APCS$+W$*APCS$IT5     ;MOVE FWORD TO BUS1
     06E20 EE706C18 (05065)          LAF     R7,APCS$+W$*APCS$IT5     ;GET ADDR OF BUFFER START
     06E22 9C70000A (05066)          ADDIR   R7,5*W$                  ;ADD HW OFFSET TO DESIRED SAMPLE
     06E24 EF706C18 (05067)          SAF     R7,APCS$+W$*APCS$IT5     ;STORE ADDR OF DESIRED SAMPLE
                    (05068) ;
                    (05069) ; T(0) (OUTPUT)...
     06E26 C04C0582 (05070)          MOVMRL  R4,BCT$BA(R6)            ;1BIT LEN., 2BIT BUS#, 17BIT BA.
     06E28 F0706C76 (05071)          MOVMR   R7,APCS$+W$*APCS$OT0     ;GET LEFT HWORD OF INSTR
     06E2A 564EFFF0 (05072)          IORKR   R4,R7,$FFF0              ;"OR" IN HIGH 4 BITS
     06E2C 84406C76 (05073)          MOVRML  R4,APCS$+W$*APCS$OT0     ;MOVE FWORD TO BUS1
                    (05074) ;
     06E2E 80000F63 (05075)          JMP     AP$BNDR0
```

```
                   ' APU3 - GTHR(Y,A,U,B,V,W)
                   ; KFIELD 9/80
                   ;
                   ; RUNS WITH GTHRS$
                   ;
                   ; FUNCTION:
                   ;       GATHER CODED TRANSMISSION DATA INTO OUTPUT BUFFER Y.
                   ;       Y <=    (216) CODED APC RESIDUAL SAMPLES
                   ;               (  1) CODED ENERGY
                   ;               (  3) CODED DELTA GAINS
                   ;               (  1) CODED PITCH LAG
                   ;               (  3) CODED PITCH PREDICTOR COEFFS
                   ;               (  6) CODED SPECTRAL PREDICTOR COEFFS
                   ;               -----
                   ;               (230)
                   ;
                   ;       ON FUNCTION CALL, Y(0-215) IS EXPECTED TO CONTAIN
                   ;       216 CODED APC RESIDUAL SAMPLES.
                   ;       ALL PARAMETERS (BUFFERS & SCALARS) ARE TREATED AS FXD,LNG
                   ;       (FOR REAL SCALARS, THIS IMPLIES USING THE LEFT HWORD OF
                   ;        THE 32 BIT SCALAR.)
                   ;
                   ;       Y(216)      <= SA
                   ;       Y(217-219)  <= U(0-2)
                   ;       Y(220)      <= SB
                   ;       Y(221-223)  <= V(0-2)
                   ;       Y(224-229)  <= W(0-5)
                   ;
                   ; INPUT SEQUENCE:
                   ;       SA, U(0), U(1), U(2),
                   ;       SB, V(0), V(1), V(3),
                   ;       W(0), ..., W(5), [FI]
                   ;
                   ; OUTPUT SEQUENCE:
                   ;       Y(216), ..., Y(229), [EO]
                   ;
                           EVEN
    06E30 0000             DATA    GTHRU$SA
    06E31 0004             DATA    GTHRU$SZ
                   ;
                   GTHRU$  BEGIN   APU(GTHRU)
                   GTHRU$SA:
A00 06E32 00FC0000         MOV(IQA,OQ) \ NOP            ;COPY INPUT TO OUTPUT
A01 06E34 901C0000         JUMPC(GTHRU$SA,EO)           ;LOOP TIL DONE
                   ;
A02 06E36 20322032         CLEAR(RA)
A03 06E38 10000000         JUMP(0)
          00000004 GTHRU$SZ = $A-GTHRU$SA
    06E3A                  END
```

```
                   (05125) ' APS3 - GTHR(Y,A,U,B,V,W)
                   (05126) ; KFIELD 9/80
                   (05127) ;
                   (05128) ; INPUT SEQUENCE:
                   (05129) ;       SA, U(0), U(1), U(2),
                   (05130) ;       SB, V(0), V(1), V(3),
                   (05131) ;       W(0), ..., W(5), [PI]
                   (05132) ;
                   (05133) ; OUTPUT SEQUENCE:
                   (05134) ;       Y(216), ..., Y(229), [EO]
                   (05135) ;
                   (05136) ; ALL DATA ITEMS ARE FXD,LNG (TE FORMAT)
                   (05137) ;
                   (05138)         EVEN
    06E3A 00006E84 (05139)         ADDR    GTHRS$I               ;PTR TO CONS INS BLOCK
    06E3C 00006E46 (05140)         ADDR    GTHRS$+2*GTHRS$S      ;SCALAR BLOCK
    06E3E 0002     (05141)         DATA    2                     ;# OF SCALARS
    06E3F 0056     (05142)         DATA    GTHRS$Z               ;MODULE SIZE
    06E40 00006E7C (05143)         ADDR    GTHRS$A               ;PTR TO CHAIN ANCHOR
                   (05144) ;
                   (05145) GTHRS$: BEGIN   APS(GTHRS)
A00 06E42 00201740 (05146)         JSN(GTHRS$O,P2)               ;SET UP & START OUTPUT PGN
A01 06E44 02300030 (05147)         SET(RO)
                   (05148) ;
                   (05149) ; INPUT PGN
                   (05150) ;
                   (05151) GTHRS$S:
A02 06E46 05420000 (05152)         LOAD(BR0,M$$(1),TE)           ;GEN SA ADDR
A03 06E48 06420000 (05153)         LOAD(BR0,M$$(1))              ;LOAD SB ADDR
                   (05154) ;
                   (05155) ; GEN U(0-2)
                   (05156) ;
A04 06E4A 09501000 (05157)         LOAD(BR1,[1],TE)              ;GEN U(0)
A05 06E4C 0A600000 (05158)         LOAD(BR2,M$$)                 ;DUMMY
A06 06E4E 0C600000 (05159)         LOAD(BR2,M$$)                 ;DUMMY
                   (05160) ;
A07 06E50 0F1A9006 (05161)         ADD(BR1,[9],TE)               ;GEN U(1)
A08 06E52 111A9002 (05162)         ADD(BR1,[9],TE)               ;GEN U(2)
                   (05163) ;
                   (05164) ; GEN SB
                   (05165) ;
A09 06E54 13090011 (05166)         MOVB(BR0,BR0,TE)              ;GEN SB
                   (05167) ;
                   (05168) ; GEN V(0-2)
                   (05169) ;
A0A 06E56 15502004 (05170)         LOAD(BR1,[2],TE)              ;GEN V(0)
A0B 06E58 16600000 (05171)         LOAD(BR2,M$$)                 ;DUMMY
A0C 06E5A 18600000 (05172)         LOAD(BR2,M$$)                 ;DUMMY
                   (05173) ;
A0D 06E5C 1B1AA006 (05174)         ADD(BR1,[10],TE)              ;GEN V(1)
A0E 06E5E 1D1AA002 (05175)         ADD(BR1,[10],TE)              ;GEN V(2)
                   (05176) ;
                   (05177) ; GEN W(0-5)
```

APS3 - GTWR(Y,A,U,B,V,W)

```
                   (05178) ;                                      ;LOAD W(0) ADDR
A0F 06E60 1E503002 (05179)        LOAD(BR1,[3])
A10 06E62 20600000 (05180)        LOAD(BR2,N$$)                   ;DUMMY
A11 06E64 22120000 (05181)        SUB(BR1,N$$)                    ;LOAD W(-1) ADDR
                   (05182) ;                                      ;GEN 6 SAMPLES
A12 06E66 24600005 (05183)        LOAD(BR2,6-1)                   ;GEN W ELEMENT ADDR
A13 06E68 271A0000 (05184) 01:    ADD(BR1,[11],TE)
A14 06E6A 20291301 (05185)        SUBL(BR2,1),JUMPP(01)
                   (05186) ;                                      ;DONE WITH INPUT
A15 06E6C 2A200031 (05187)        CLEAR(R1)
A16 06E6E 2C000020 (05188)        NOP(0)
                   (05189)        EJECT
```

APS3 - GTHR(Y,A,U,B,V,W)

```
                    (05190) ;
                    (05191) ; OUTPUT PGM
                    (05192) ;
                    (05193) GTHRS$O:
A17 06E70 2E300032 (05194)          SET(RA)                         ;START APU
                    (05195) ;
                    (05196) ; GEN Y(216-229)
                    (05197) ;
A18 06E72 305F000A (05198)          LOAD(BW1,C03)                   ;LOAD Y(0) ADDR
A19 06E74 32600000 (05199)          LOAD(BW2,M$$)                   ;DUMMY
A1A 06E76 34120000 (05200)          SUB(BW1,M$$)                    ;LOAD Y(-1) ADDR
                    (05201) ;
A1B 06E78 361A00D8 (05202)          ADD(BW1,216)                    ;LOAD Y(215) ADDR
A1C 06E7A 3060000D (05203)          LOAD(BW2,14-1)                  ;GEN 14 SAMPLES
                    (05204) ;
A1D 06E7C 3A1A000A (05205) @2:      ADD(BW1,C03,TE)                 ;GEN Y ELEM ADDR
A1E 06E7E 3C211D01 (05206)          SUBL(BW2,1),JUMPP(@2)
                    (05207) ;
A1F 06E80 3E200030 (05208)          CLEAR(RO)                       ;DONE WITH OUTPUT
A20 06E82 40000020 (05209)          NOP(0)
                    (05210) ;
    00006E7C        (05211) GTHRS$A=@C                              ;CHAIN ANCHOR
                    (05212) ;
    06E84           (05213)          END
                    (05214) ;
    06E84 00000000 (05215) GTHRS$I DATA    16F'0.0'
     ...
                    (05216) ;
    00000056        (05217) GTHRS$Z=@L-GTHRS$                       ;MODULE SIZE
```

```
                        (05218) ' APU3 - THIST(Y,U,V,W,R,S)   SET UP XMITTER FRAME HISTORIES
                        (05219) ; KFIELD 7/80
                        (05220) ;
                        (05221) ; RUNS WITH APS3-THIST (THS$)
                        (05222) ;
                        (05223) ; Y(0)-Y(UBS-1) <= U(0)-U(UBS-1)
                        (05224) ; V(0)-V(WBS-1) <= W(0)-W(WBS-1)
                        (05225) ; R(0)-R(SBS-1) <= S(0)-S(SBS-1)
                        (05226) ;
                        (05227) ; (REQUIRES SPECIAL BINDING TO ACCOMODATE R & S BUFFERS)
                        (05228) ;
                        (05229) ; INPUT SEQUENCE:
                        (05230) ;       U(0),...,U(UBS-1),
                        (05231) ;       W(0),...,W(WBS-1),
                        (05232) ;       S(0),...,S(SBS-1), [PI]
                        (05233) ;
                        (05234) ; OUTPUT SEQUENCE:
                        (05235) ;       Y(0),...,Y(UBS-1),
                        (05236) ;       V(0),...,V(WBS-1),
                        (05237) ;       R(0),...,R(SBS-1), [EO]
                        (05238) ;
                        (05239)         EVEN
    06E98 0000          (05240)         DATA    THU$SA
    06E99 0004          (05241)         DATA    THU$SZ
                        (05242) ;
                        (05243) THU$:   BEGIN   APU(THISTU)
A00 06E9A 0BFC0000      (05244) THU$SA: MOV(TQA,OQ) \ NOP       ;COPY INPUT TO OUTPUT
A01 06E9C 981C0000      (05245)         JUMPC(THU$SA,EO)        ;LOOP TIL DONE
                        (05246) ;
A02 06E9E 20322032      (05247)         CLEAR(RA)
A03 06EA0 10000000      (05248)         JUMP(0)
          00000004      (05249) THU$SZ = $A-THU$SA
    06EA2               (05250)         END
```

```
                    (05251) * APS3 - TWIST(Y,U,V,W,R,S)   APS PROGRAM FOR TWIST
                    (05252) ; KFIELD 7/80
                    (05253) ;
                    (05254) ; BINDING DONE BY SHMSTM TO ACCOMODATE R & S BUFFERS
                    (05255) ;
                    (05256) ; INPUT SEQUENCE:
                    (05257) ;       U(0),...,U(UBS-1),
                    (05258) ;       W(0),...,W(WBS-1),
                    (05259) ;       S(0),...,S(SBS-1), [P1]
                    (05260) ;
                    (05261) ; OUTPUT SEQUENCE:
                    (05262) ;       Y(0),...,Y(UBS-1),
                    (05263) ;       V(0),...,V(WBS-1),
                    (05264) ;       R(0),...,R(SBS-1), [R0]
                    (05265) ;
                    (05266)         EVEN
     06EA2 00000000 (05267)         ADDR    0       ;(PTR TO CONS TWS BLOCK)
     06EA4 00000000 (05268)         ADDR    0       ;(SCALAR ADDRESS)
     06EA6 0000     (05269)         DATA    0       ;(NUMBER OF SCALARS)
     06EA7 004A     (05270)         DATA    TWS$Z   ;MODULE SIZE
     06EA8 00000000 (05271)         ADDR    TWS$C   ;PTR TO CHAIN ANCHOR
                    (05272) ;
                    (05273) TWS$    BEGIN   APS(TWISTS)
 A00 06EAA 00201340 (05274)         JSB(TWS$O,P2)   ;SET UP AND START OUTPUT PROGRAM
 A01 06EAC 02300030 (05275)         SET(R0)
                    (05276) ;
                    (05277) ; INPUT PROGRAM
                    (05278) ;       BR0: SAMPLE ADDRESSES
                    (05279) ;       BR1: BUFFER SIZES
                    (05280) ;
                    (05281) TWS$I:
                    (05282) ;
                    (05283) ; DO U BUFFER...
 A02 06EAE 04400000 (05284) TWS$UBA:        LOAD(BR0,M$$)   ;U BASE (WHEN BOUND)
 A03 06EB0 06500000 (05285) TWS$UBS1:       LOAD(BR1,M$$)   ;U SIZE-1 (WHEN BOUND)
 A04 06EB2 08020000 (05286) TWS$USI1:       SUB(BR0,M$$)    ;U INCR (WHEN BOUND)
                    (05287) ;
                    (05288) TWS$USI2:
 A05 06EB4 0A010000 (05289) 01              ADD(BR0,M$$,TF)         ;GEN U ADDRS (WHEN BOUND)
 A06 06EB6 0C190501 (05290)                 SUBL(BR1,1),JUMPP(01)   ;LOOP UBS TIMES
                    (05291) ;
                    (05292) ; DO W BUFFER...
 A07 06EB8 0E400000 (05293) TWS$WBA:        LOAD(BR0,M$$)   ;W BASE (WHEN BOUND)
 A08 06EBA 10500000 (05294) TWS$WBS1:       LOAD(BR1,M$$)   ;W SIZE-1 (WHEN BOUND)
 A09 06EBC 12020000 (05295) TWS$WSI1:       SUB(BR0,M$$)    ;W INCR (WHEN BOUND)
                    (05296) ;
                    (05297) TWS$WSI2:
 A0A 06EBE 14010000 (05298) 02              ADD(BR0,M$$,TF)         ;GEN W ADDRS (WHEN BOUND)
 A0B 06EC0 16190A01 (05299)                 SUBL(BR1,1),JUMPP(02)   ;LOOP WBS TIMES
                    (05300) ;
                    (05301) ; DO S BUFFER...
 A0C 06EC2 18400000 (05302) TWS$SBA:        LOAD(BR0,M$$)   ;S BASE (WHEN BOUND)
 A0D 06EC4 1A500000 (05303) TWS$SBS1:       LOAD(BR1,M$$)   ;S SIZE-1 (WHEN BOUND)
```

```
A0E 06EC6 1C020000 (05304) TWS$SSI1:       SUB(BW0,M$$)    ;S INCR (WHEN BOUND)
                   (05305) ;
                   (05306) TWS$SSI2:
A0F 06EC8 1E0A0000 (05307) 03              ADD(BW0,M$$,TF)        ;GEN S ADDRS (WHEN BOUND)
A10 06ECA 20190F01 (05308)                 SUBL(BW1,1),JUMPP(03)  ;LOOP SBS TIMES
                   (05309) ;
A11 06ECC 22200031 (05310)         CLEAR(RI)       ;DONE WITH INPUT
A12 06ECE 24000020 (05311)         NOP(0)
                   (05312) ;
                   (05313) ;
                   (05314) ; OUTPUT PROGRAM
                   (05315) ;       BW0: SAMPLE ADDRESSES
                   (05316) ;       BW1: BUFFER SIZES
                   (05317) ;
A13 06ED0 26300032 (05318) TWS$O:  SET(RA)
                   (05319) ;
                   (05320) ; DO Y BUFFER...
A14 06ED2 28400000 (05321) TWS$YBA:        LOAD(BW0,M$$)   ;Y BASE (WHEN BOUND)
A15 06ED4 2A500000 (05322) TWS$UBS2:       LOAD(BW1,M$$)   ;U SIZE-1 (WHEN BOUND)
A16 06ED6 2C020000 (05323) TWS$YSI1:       SUB(BW0,M$$)    ;Y INCR (WHEN BOUND)
                   (05324) ;
                   (05325) TWS$YSI2:
A17 06ED8 2E0A0000 (05326) 04              ADD(BW0,M$$,TF)        ;GEN Y ADDRS (WHEN BOUND)
A18 06EDA 30111701 (05327)                 SUBL(BW1,1),JUMPP(04)  ;LOOP UBS TIMES
                   (05328) ;
                   (05329) ; DO V BUFFER...
A19 06EDC 32400000 (05330) TWS$VBA:        LOAD(BW0,M$$)   ;V BASE (WHEN BOUND)
A1A 06EDE 34500000 (05331) TWS$WBS2:       LOAD(BW1,M$$)   ;W SIZE-1 (WHEN BOUND)
A1B 06EE0 36020000 (05332) TWS$VSI1:       SUB(BW0,M$$)    ;V INCR (WHEN BOUND)
                   (05333) ;
                   (05334) TWS$VSI2:
A1C 06EE2 380A0000 (05335) 05              ADD(BW0,M$$,TF)        ;GEN V ADDRS (WHEN BOUND)
A1D 06EE4 3A111C01 (05336)                 SUBL(BW1,1),JUMPP(05)  ;LOOP WBS TIMES
                   (05337) ;
                   (05338) ; DO R BUFFER...
A1E 06EE6 3C400000 (05339) TWS$RBA:        LOAD(BW0,M$$)   ;R BASE (WHEN BOUND)
A1F 06EE8 3E500000 (05340) TWS$SBS2:       LOAD(BW1,M$$)   ;S SIZE-1 (WHEN BOUND)
A20 06EEA 40020000 (05341) TWS$RSI1:       SUB(BW0,M$$)    ;R INCR (WHEN BOUND)
                   (05342) ;
                   (05343) TWS$RSI2:
A21 06EEC 420A0000 (05344) 06              ADD(BW0,M$$,TF)        ;GEN R ADDRS (WHEN BOUND)
A22 06EEE 441121B1 (05345)                 SUBL(BW1,1),JUMPP(06)  ;LOOP SBS TIMES
                   (05346) ;
                   (05347) ;
A23 06EF0 46200030 (05348)         CLEAR(RO)       ;DONE WITH OUTPUT
A24 06EF2 48000020 (05349)         NOP(0)
          00000000 (05350) TWS$C=0C
    06EF4          (05351)         END
                   (05352) ;
          0000004A (05353) TWS$Z=0L-TWS$
```

```
PAGE  120:  [BBND]<BCA16>BBM16M.MSQ.2, 29-Dec-80 14:30:40, Ed: WOLF
          SBM$TH  SPECIAL BINDING MODULE FOR THIST

                    (05354) * SBM$TH  SPECIAL BINDING MODULE FOR THIST
                    (05355) ;
                    (05356) ;        PCB FORMAT (16 BIT FORMAT SHOWN)
                    (05357) ;
                    (05358) ;      WORD      LEFT BYTE          RIGHT BYTE
                    (05359) ;      ----      ---------          ----------
                    (05360) ;       0:        PTR TO NEXT PCB AND FUNCTION LIST FLAG (LSB)
                    (05361) ;       1:          201                CONTROL BITS
                    (05362) ;       2:           T                  U
                    (05363) ;       3:           V                  W
                    (05364) ;       4:           R                  S
                    (05365) ;       5:           0                  0
                    (05366) ;
                    (05367) ;
                    (05368) SBM$TH:
06EF4 C63400E8 (05369)          PUSHMIL R3,APDT$ORG(R2)      STACK APU MODULE BUS ORIGIN
06EF6 9071FFFE (05370)          MOVIR R7,-W$
06EF8 C607FFFE (05371)          PUSHMIL R3,0-W$(R3)          STACK APU MODULE ST ADDR AND SZ
06EFA C63400EA (05372)          PUSHMIL R3,APDT$ORG+W$(R2)   STACK APS MODULE BUS ORIGIN
06EFC C63400EC (05373)          PUSHMIL R3,APDT$ORG+2*W$(R2) STACK CSPU SUPPORT MODULE
06EFE 2771     (05374)          DECR R7,1
06EFF 0000     (05375)          EVEN
06F00 C407FFFC (05376)          PUSHMI R3,0-2*W$(R3)         APS MODULE SIZE
06F02 9047FFFD (05377)          MOVIR R4,-3*W$(R3)           GET SPECIAL SUPPORT SEMAPHORE
06F04 7040FF00 (05378)          MOVWK R4,R4,MSK$LBYT
06F06 3C40     (05379)          LRS R4,8
06F07 3430     (05380)          PUSHXI R3,R4                 STACK SPECIAL SUPPORT SEMAPHORE
06F08 4052     (05381)          MOVRR R5,RI                  POINT TO FIRST SCALAR ID
06F09 2651     (05382)          INCR R5,H$
06F0A 704A00FF (05383)          MOVWK R4,R5,MSK$RBYT         EXTRACT SCALAR A IDENTIFIER
06F0C 3630     (05384)          PUSHXIL R3,R4                STACK SCALAR A IDENTIFIER
06F0D C6300792 (05385)          PUSHMIL R3,CLR$GBG1          RESET G FLAGS
06F0F 2632     (05386)          INCR R3,2
06F10 C63007BA (05387)          PUSHMIL R3,ZERO              SET FOR NO PBF
06F12 9E300011 (05388)          SUBIR R3,AP$LLOC
                    (05389) ;
                    (05390) ;
                    (05391) ; V BUFFER BINDING:
                    (05392) ;
                    (05393) ; V BASE...
06F14 F0620001 (05394)         MOVMR    R6,W$(R1)
06F16 506CFF00 (05395)         MOVKR    R6,R6,MSK$LBYT   ;GET BID
06F18 3C67     (05396)         LRS      R6,7             ;CONVERT TO BCT INDEX
06F19 0000     (05397)         EVEN
06F1A C04C0502 (05398)         MOVMRL   R4,BCT$BA(R6)   ;1BIT LEN., 2BIT BUSB, 17BIT BA.
06F1C F0706ED2 (05399)         MOVMR    R7,THS$+W$*THS$VBA       ;GET LEFT HWORD OF INSTR
06F1E 564FFFF0 (05400)         IORKR    R4,R7,$FFF0      ;"OR" IN HIGH 4 BITS
06F20 B4406ED2 (05401)         MOVRML   R4,THS$+V$*THS$VBA       ;MOVE FWORD TO BUSI
                    (05402) ;
                    (05403) ; VSI...
06F22 F04C0604 (05404)         MOVMR    R4,BCT$AD(R6)   ;SAMPLE INCREMENT
06F24 EF406ED6 (05405)         SAF      R4,THS$+W$*THS$VSI1
06F26 EF406ED8 (05406)         SAF      R4,THS$+V$*THS$VSI2
```

```
                     (05407) ;
                     (05408) ; W BUFFER BINDING:
                     (05409) ;
                     (05410) ; WBA...
06F28 F0620001 (05411)          MOVHR   R6,H$(R1)
06F2A 506C00FF (05412)          MOVKR   R6,R6,MSK$RBYT  ;GET BID
06F2C 3A61     (05413)          LLS     R6,1             ;CONVERT TO BCT INDEX
06F2D 0000     (05414)          EVEN
06F2E C04C0502 (05415)          MOVHRL  R4,BCT$BA(R6)   ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06F30 F0706EAE (05416)          MOVHR   R7,THS$+W$*THS$UBA      ;GET LEFT HWORD OF INSTR
06F32 564FFFF0 (05417)          IORKR   R4,R7,$FFF0     ;'OR' IN HIGH 4 BITS
06F34 84406EAE (05418)          MOVRHL  R4,THS$+W$*THS$UBA      ;MOVE FWORD TO BUS1
                     (05419) ;
                     (05420) ; WBS...
06F36 F04C0605 (05421)          MOVHR   R4,H$+BCT$AD(R6)        ;SIZE-1
06F38 E0406EB1 (05422)          MOVRH   R4,THS$+W$*THS$UBS1+H$
06F3A E0406ED5 (05423)          MOVRH   R4,THS$+W$*THS$UBS2+H$
                     (05424) ;
                     (05425) ; WSI...
06F3C F04C0604 (05426)          MOVHR   R4,BCT$AD(R6)           ;SAMPLE INCREMENT
06F3E EF406EB2 (05427)          SAF     R4,THS$+W$*THS$USI1
06F40 EF406ED4 (05428)          SAF     R4,THS$+W$*THS$USI2
                     (05429) ;
                     (05430) ; V BUFFER BINDING:
                     (05431) ;
                     (05432) ; V BASE...
06F42 F0620002 (05433)          MOVHR   R6,2*H$(R1)
06F44 506CFF00 (05434)          MOVKR   R6,R6,MSK$LBYT  ;GET BID
06F46 3C67     (05435)          LRS     R6,7             ;CONVERT TO BCT INDEX
06F47 0000     (05436)          EVEN
06F48 C04C0502 (05437)          MOVHRL  R4,BCT$BA(R6)   ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06F4A F0706EDC (05438)          MOVHR   R7,THS$+W$*THS$VBA      ;GET LEFT HWORD OF INSTR
06F4C 564FFFF0 (05439)          IORKR   R4,R7,$FFF0     ;'OR' IN HIGH 4 BITS
06F4E 84406EDC (05440)          MOVRHL  R4,THS$+W$*THS$VBA      ;MOVE FWORD TO BUS1
                     (05441) ;
                     (05442) ; VSI...
06F50 F04C0604 (05443)          MOVHR   R4,BCT$AD(R6)   ;SAMPLE INCREMENT
06F52 EF406EE0 (05444)          SAF     R4,THS$+W$*THS$VSI1
06F54 EF406EE2 (05445)          SAF     R4,THS$+W$*THS$VSI2
                     (05446) ;
                     (05447) ; W BUFFER BINDING:
                     (05448) ;
                     (05449) ; WBA...
06F56 F0620002 (05450)          MOVHR   R6,2*H$(R1)
06F58 506C00FF (05451)          MOVKR   R6,R6,MSK$RBYT  ;GET BID
06F5A 3A61     (05452)          LLS     R6,1             ;CONVERT TO BCT INDEX
06F5B 0000     (05453)          EVEN
06F5C C04C0502 (05454)          MOVHRL  R4,BCT$BA(R6)   ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06F5E F0706EB0 (05455)          MOVHR   R7,THS$+W$*THS$WBA      ;GET LEFT HWORD OF INSTR
06F60 564FFFF0 (05456)          IORKR   R4,R7,$FFF0     ;'OR' IN HIGH 4 BITS
06F62 84406EB0 (05457)          MOVRHL  R4,THS$+W$*THS$WBA      ;MOVE FWORD TO BUS1
                     (05458) ;
                     (05459) ; WBS...
```

SBMSTN SPECIAL BINDING MODULE FOR THIST

```
06F64 F04C0605 (05460)        MOVNR   R4,H$+BCT$AD(R6)          ;SIZE-1
06F66 E0406EBD (05461)        MOVRM   R4,THS$+W$*THS$WBS1+H$
06F68 E0406EDF (05462)        MOVRM   R4,THS$+W$*THS$WBS2+H$
               (05463) ;
               (05464) ; WSI...
06F6A F04C0604 (05465)        MOVNR   R4,BCT$AD(R6)             ;SAMPLE INCREMENT
06F6C EF406EBC (05466)        SAF     R4,THS$+W$*THS$WSI1
06F6E EF406EBE (05467)        SAF     R4,THS$+W$*THS$WSI2
               (05468) ;
               (05469) ; R BUFFER BINDING:
               (05470) ;
               (05471) ; R BASE...
06F70 F0620003 (05472)        MOVNR   R6,3*H$(R1)
06F72 506CFF00 (05473)        MOVKR   R6,R6,MSK$LBYT  ;GET BID
06F74 3C67     (05474)        LRS     R6,7                ;CONVERT TO BCT INDEX
06F75 0000     (05475)        EVEN
06F76 C04C0502 (05476)        MOVNRL  R4,BCT$BA(R6)   ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06F78 F0706EE6 (05477)        MOVNR   R7,THS$+W$*THS$RBA        ;GET LEFT HWORD OF INSTR
06F7A 564FFFF0 (05478)        IORKR   R4,R7,$FFF0      ;"OR" IN HIGH 4 BITS
06F7C 84406EE6 (05479)        MOVRNL  R4,THS$+W$*THS$RBA        ;MOVE FWORD TO BUS1
               (05480) ;
               (05481) ; RSI...
06F7E F04C0604 (05482)        MOVNR   R4,BCT$AD(R6)   ;SAMPLE INCREMENT
06F80 EF406EEA (05483)        SAF     R4,THS$+W$*THS$RSI1
06F82 EF406EEC (05484)        SAF     R4,THS$+W$*THS$RSI2
               (05485) ;
               (05486) ; S BUFFER BINDING:
               (05487) ;
               (05488) ; SBA...
06F84 F0620003 (05489)        MOVNR   R6,3*H$(R1)
06F86 506C00FF (05490)        MOVKR   R6,R6,MSK$RBYT  ;GET BID
06F88 3A61     (05491)        LLS     R6,1                ;CONVERT TO BCT INDEX
06F89 0000     (05492)        EVEN
06F8A C04C0502 (05493)        MOVNRL  R4,BCT$BA(R6)   ;1BIT LEN., 2BIT BUS#, 17BIT BA.
06F8C F0706EC2 (05494)        MOVNR   R7,THS$+W$*THS$SBA        ;GET LEFT HWORD OF INSTR
06F8E 564FFFF0 (05495)        IORKR   R4,R7,$FFF0      ;"OR" IN HIGH 4 BITS
06F90 84406EC2 (05496)        MOVRNL  R4,THS$+W$*THS$SBA        ;MOVE FWORD TO BUS1
               (05497) ;
               (05498) ; SBS...
06F92 F04C0605 (05499)        MOVNR   R4,H$+BCT$AD(R6)          ;SIZE-1
06F94 E0406EC5 (05500)        MOVRM   R4,THS$+W$*THS$SBS1+H$
06F96 E0406EE9 (05501)        MOVRM   R4,THS$+W$*THS$SBS2+H$
               (05502) ;
               (05503) ; SSI...
06F98 F04C0604 (05504)        MOVNR   R4,BCT$AD(R6)             ;SAMPLE INCREMENT
06F9A EF406EC6 (05505)        SAF     R4,THS$+W$*THS$SSI1
06F9C EF406EC8 (05506)        SAF     R4,THS$+W$*THS$SSI2
               (05507) ;
               (05508) ;
06F9E 00000F63 (05509)        JMP     AP$BNDR0
```

```
                   (05510) * APU3 - SCLRES(V,A,U,V)
                   (05511) * WHR 9/80
                   (05512) *
                   (05513) *  ROUTINE TO COMPUTE QUANTIZER SCALE FACTORS
                   (05514) *  AND SCALE RESIDUAL
                   (05515) *
                   (05516) *  INPUTS:
                   (05517) *       V(1),...,V(216) = RECEIVED RESIDUAL SAMPLES
                   (05518) *       U(1),U(2),U(3)  = RECEIVED DELTA GAINS (DG(I))
                   (05519) *              SA       = RECEIVED OVERALL GAIN
                   (05520) *
                   (05521) *  OUTPUT:
                   (05522) *       V(1),...,V(216) = SCALED RESIDUAL SAMPLES
                   (05523) *
                   (05524) *  PROCEDURE:
                   (05525) *       COMPUTE GFAC(I)=SA*U(I),I=1,2,3
                   (05526) *       SCALE RESIDUAL V(I)=V(I)*GFAC(J)
                   (05527) *              J=1 FOR I=1,...,72
                   (05528) *              J=2 FOR I=73,...,144
                   (05529) *              J=3 FOR I=145,...,216
                   (05530) *
                   (05531) *  INPUT SEQUENCE
                   (05532) *       IQ: SA,U(1),U(2),U(3),V(1),...,V(216)
                   (05533) *  OUTPUT SEQUENCE
                   (05534) *       DQ: DUM,DUM,V(1),...,V(216)
                   (05535) *
                   (05536) *  APU REGISTER USAGE
                   (05537) *  ADDER REGISTERS NOT USED
                   (05538) *       M0     V(I)
                   (05539) *       M1     DG(1)
                   (05540) *       M2     DG(2)
                   (05541) *       M3     DG(3)
                   (05542) *       M4
                   (05543) *       M5     SA
                   (05544) *       M6     GFAC
                   (05545) *
                   (05546) *
                   (05547) *  SCL - APU - INITIALIZATION
                   (05548) *
                   (05549)         EVEN
    06FA0 0000     (05550)         DATA SCL$SA                    ;APU START ADDR
    06FA1 0019     (05551)         DATA SCL$SZ                    ;SIZE
                   (05552) *
                   (05553) SCL$:   BEGIN   APU(SCLRES)
A00 06FA2 00ED00ED (05554) SCL$SA: MOV(IQA,M5)                    ;M5= SA =SQRT(ENERGY)
A01 06FA4 00E900E9 (05555)         MOV(IQA,M1)                    ;M1= U(1) = DG(1)
A02 06FA6 00EA00EA (05556)         MOV(IQA,M2)                    ;M2= U(2) = DG(2)
A03 06FA8 00EB00EB (05557)         MOV(IQA,M3)                    ;M3= U(3) = DG(3)
                   (05558) * 1ST SEGMENT
A04 06FAA 04A004A0 (05559)         MUL(M1,M5)                     ;P=GFAC(1)
A05 06FAC 3000000C (05560)         CALL(SCL$M)                    ;SCALE V
                   (05561) * 2ND SEGMENT
A06 06FAE 05200520 (05562)         MUL(M2,M5)                     ;P=GFAC(2)
```

```
A07 06FB0 3000000C (05563)          CALL(SCL$M)                ;SCALE V
                   (05564) * 3RD SEGMENT
A08 06FB2 05A005A0 (05565)          MUL(M3,M5)                 ;P=GFAC(3)
A09 06FB4 3F00000C (05566)          CALL(SCL$M)                ;SCALE V
                   (05567) * DONE
A0A 06FB6 20322032 (05568)          CLEAR(RA)                  ;HALT APU
A0B 06FB8 10000000 (05569)          JUMP(0)
                   (05570) *
                   (05571) *
                   (05572) *  SUBROUTINE TO SCALE RESIDUAL
A0C 06FBA 00AE00AE (05573) SCL$M:  MOV(P,M6)                  ;M6=GFAC
A0D 06FBC 00000000 (05574) SCL$M1: NOP
A0E 06FBE 00E00000 (05575)          MOV(IQA,M8)\NOP            ;M8=V(I)
A0F 06FC0 045C00E0 (05576)          MOV(DQ),MUL(M8,M6)\MOV(IQA,M8)  ;DQ=V(I-2),P=V(I)\M8=V(I+1)
A10 06FC2 0000045C (05577)          NOP\MOV(DQ),MUL(M8,M6)     ;NOP\DQ=V(I-1),P=V(I+1)
A11 06FC4 00000000 (05578)          NOP
A12 06FC6 00000000 (05579)          NOP
A13 06FC8 9016000D (05580)          JUMPC(SCL$M1,FWI)          ;CONTINUE IF MORE INPUT
A14 06FCA 00000000 (05581)          NOP
                   (05582) *  CLEAN UP
A15 06FCC 000C0000 (05583)          MOV(P,DQ)\NOP              ;DQ=V
A16 06FCE 0000000C (05584)          NOP\MOV(P,DQ)              ;DQ=V
A17 06FD0 20372037 (05585)          CLEAR(WI)                  ;RESTART APS
A18 06FD2 A000A000 (05586)          RETURN
                   (05587) *
        00000019 (05588) SCL$SZ=$A-SCL$SA                ;COMPUTE SIZE
                   (05589) *
    06FD4          (05590)          END
                   (05591) *
```

```
                        * APS3 - SCLRES(V,A,U,V)
                        * WHR 9/80
                        *
                        *  APS ROUTINE FOR SCL
                        *
                        *  INPUTS:
                        *       IQ: SA,U(1),U(2),U(3),V(1),...,V(216)
                        *  OUTPUT:
                        *       OQ: DUM,DUM,Y(1),...,Y(216)
                        *
                        *  INPUT PROGRAM REGISTER USAGE
                        *       BR0     SA,U,V
                        *       BR1     SIZE (UNUSED)
                        *       BR2     COUNTER(72)
                        *       BR3     COUNTER(3)
                        *
                        *  HEADER BLOCK
                                EVEN
    06FD4 00007028              ADDR SCLS$I                ;PTR TO CONSTRUCTED INSTRUCTION  BLOCK
    06FD6 00006FE0              ADDR SCLS$+2*SCLS$S        ;PTR TO SCALER BLOCK
    06FD8 0001                  DATA 1                     ;SET NUMBER OF SCALERS
    06FD9 0050                  DATA SCLS$Z                ;SET MODULE SIZE
    06FDA 00007014              ADDR SCLS$A                ;POINTER TO CHAIN ANCHOR
                                EVEN                       ;START APS PRG ON FULL WORDS
                        *
                        *  END OF HEADER BLOCK
                        *
                        * START OF APS PROGRAM
                        SCLS$:  BEGIN APS(SCLS)
                        *
A00 06FDC 00201440              JSN(SCLS$OP,P2)
A01 06FDE 02300030              SET(RO)
                        *  INPUT SCALER
A02 06FE0 04C20000      SCLS$S: LOAD(BR0,M$$(1),TF)        ;LOAD AND GEN SA ADDR(GAIN)
                        *
                        *  LOAD DELTA GAINS - U
A03 06FE2 06C01000              LOAD(BR0,[1],TF)           ;LOAD AND GEN U(1) ADDR
A04 06FE4 08500000              LOAD(BR1,M$$)              ;(UNUSED)
A05 06FF6 0A0A0000              ADD(BR0,M$$,TF)            ;GEN U(2) ADDR
A06 06FF8 0C0A9006              ADD(BR0,[9],TF)            ;GEN U(3) ADDR
                        *
                        *  LOAD RESIDUAL SAMPLES
A07 06FFA 0E402002              LOAD(BR0,[2])              ;LOAD V BA
A08 06FFC 10500000              LOAD(BR1,M$$)              ;(UNUSED)
A09 06FEE 12020000              SUB(BR0,M$$)               ;PREPARE TO INC LATER
                        *  LOAD COUNTERS IN BR2 AND BR3
A0A 06FF0 14700002              LOAD(BR3,3-1)              ;BR3=# OF SEGMENTS - 1
A0B 06FF2 16600046      SCLS$1:  LOAD(BR2,72-2)            ;BR2=# OF SAMPLES/SEG - 2
                        *  LOOP FOR SEGMENT SAMPLES INPUT
A0C 06FF4 180AA00A      SCLS$2:  ADD(BR0,[10],TF)          ;GEN V(I) ADDR
A0D 06FF6 1A290CB1              SUBL(BR2,1),JUMPP(SCLS$2)  ;DEC INDEX AND JUMP
A0E 06FF8 1C090F7A              ADDL(BR0,M$,TF),JUMP($A+1) ;DO LAST VALUE AND ENSURE SET WI
A0F 06FFA 1E300037              SET(WI)                    ;SIGNAL END OF INPUT
```

```
              APS3 - SCLRES(Y,A,U,V)

A10 06FFC 20000020 (05645)          NOP(0)
A11 06FFE 22390BB1 (05646)          SUBL(BR3,1),JUMPP(SCLS$1)     ;LOOP FOR 2ND AND 3RD SEGMENT
                   (05647) *
A12 07000 24200031 (05648)          CLEAR(RI)                     ;DONE INPUT
A13 07002 26000020 (05649)          NOP(0)
                   (05650) *
                   (05651) *
                   (05652) *  OUTPUT APS PROGRAM
                   (05653) *
                   (05654) *  REGISTER USAGE
                   (05655) *        BW0     Y BASE ADDR
                   (05656) *        BW1     Y SIZE (UNUSED),DUMMYS
                   (05657) *        BW2     COUNTER (72)
                   (05658) *        BW3     COUNTER (3)
                   (05659) *
A14 07004 28300032 (05660) SCLS$OP: SET(RA)                       ;TURN ON APU
                   (05661) *
                   (05662) *  LOAD OUTPUT SCALED RESIDUAL SAMPLES
A15 07006 2A400012 (05663)          LOAD(BW0,[0])                 ;LOAD Y BA
A16 07008 2C500000 (05664)          LOAD(BW1,M$$)                 ;(UNUSED)
A17 0700A 2E020000 (05665)          SUB(BW0,M$$)                  ;PREPARE TO INC LATER
                   (05666) *  LOAD COUNTERS IN BW2 AND BW3
A18 0700C 30700002 (05667)          LOAD(BW3,3-1)                 ;BW3= # OF SEGMENTS -1
A19 0700E 32600046 (05668) SCLS$3:  LOAD(BW2,72-2)                ;BW2= # OF SAMPLES - 2
A1A 07010 34D20794 (05669)          LOAD(BW1,DMY$(1),TF)          ;LOAD DUMMY ADDR
A1B 07012 36D20794 (05670)          LOAD(BW1,DMY$(1),TF)          ;LOAD DUMMY ADDR
A1C 07014 380A000E (05671) SCLS$4:  ADD(BW0,[0],TF)               ;GEN Y(I) ADDR
A1D 07016 3A211CB1 (05672)          SUBL(BW2,1),JUMPP(SCLS$4)     ;DEC COUNTER AND JUMP
A1E 07018 3C01003A (05673)          ADDL(BW0,N$,TF)               ;GEN LAST Y(I) OF SEGM
A1F 0701A 3E311981 (05674)          SUBL(BW3,1),JUMPP(SCLS$3)     ;JUMP FOR NEXT SEGM
A20 0701C 40200030 (05675)          CLEAR(RO)
A21 0701E 42000020 (05676)          NOP(0)
                   (05677) *
          00007014 (05678) SCLS$A=$C                              ;ASSIGN VALUE TO CHAIN ANCHOR
                   (05679) *
    07020          (05680)          END
                   (05681) *
    07020 00000000 (05682) SCLS$I DATA     6F'0.0'                ;BLOCK FOR CONSTR INSTRU
     ...
                   (05683) *
          00000050 (05684) SCLS$Z=$L-SCLS$                        ;COMPUTE BUS1 MODULE SIZE
```

```
' APU3-PITSYN(Y,A,U,V)    3-TAP PITCH SYNTHESIS FILTER
;  J. WOLF, 6/30/80
;
;  BINDS TO APS3-PITSYN
;
;
;IMPLEMENTS A 3-TAP PITCH SYNTHESIS FILTER
;  Y(N) = U(N) - V(0)*Y(N-M+1) - V(1)*Y(N-M) - V(2)*Y(N-M-1)
;                 N=0,1,...,UBS-1
;
;  BUFFER Y = FILTER OUTPUT AND PREVIOUS 216 SAMPLES OF OUTPUT,
;        WHERE THE INDEX FOR Y RANGES FROM -216 TO UBS-1.  THAT IS, THE
;        FIRST ENTRY IN Y IS "LOGICAL" Y(-216), AND THE FIRST OUTPUT
;        SAMPLE ("LOGICAL" Y(0)) IS ACTUAL Y(216).  BUFFER Y SHOULD BE
;        (AT LEAST) 216+UBS SAMPLES LONG.
;    NOTE: BUFFER Y MUST BE COMPACT.
;  REAL SCALAR A = PITCH PERIOD IN SAMPLES
;  BUFFER U = FILTER INPUT
;  BUFFER V = FILTER COEFFICIENTS
;
;WE PERFORM 2 FILTER CALCULATIONS IN THE 2 ADMS:
;LEFT:
;  Y(N)    = -V(0)*Y(N-M+1) + U(N)   - V(1)*Y(N-M) - V(2)*Y(N-M-1)
;RIGHT:
;  Y(N+1) = -V(1)*Y(N-M+1) + U(N+1) - V(2)*Y(N-M) - V(0)*Y(N-M+2)
;    BECAUSE M>13 WE NEVER HAVE TO WORRY ABOUT READING A Y-VALUE
;    BEFORE IT HAS BEEN WRITTEN.
;
;INPUT SEQUENCE:
;  (2**-15), SA, MOD. APS INSTR., V(0), V(1), V(2),
;  Y(-M+1),  U(0), U(1),   Y(-M),  Y(-M-1),  Y(-M+2),
;  Y(2-M+1), U(2), U(3),   Y(2-M), Y(2-M-1), Y(2-M+2), ...,
;  Y(N-M+1), U(N), U(N+1), Y(N-M), Y(N-M-1), Y(N-M+2), ..., [FI]
;OUTPUT SEQUENCE:
; FXD PT PITCH, MOD. APS INSTR., 2 DUMMIES, Y(1), Y(2), ..., Y(UBS) [EO]
;
;DURATION:
;  18 APU CYCLES = ~1.8 USEC PER 2 OUTPUT POINTS
;  8 MEM REFS = ~1.44 USEC PER 2 OUTPUT POINTS (BUS 3)
;
;APU REGISTER USAGE:
;        LEFT           RIGHT
; A0     SUM            SUM
; A1     V*Y            V*Y
; A2     U(N)           U(N+1)
; M0     Y(N-M+1)       Y(N-M+1)
; M1     Y(N-M)         Y(N-M)
; M2     Y(N-M-1)       Y(N-M+2)
; M4     V(0)           V(1)
; M5     V(1)           V(2)
; M6     V(2)           V(0)
;
         EVEN
```

```
PAGE  120:  [BBND]<DCA16>BBN16M.MSD.2, 29-Dec-80 14:30:40, Ed: WOLF
                APU3-PITSYN(Y,A,U,V)     3-TAP PITCH SYNTHESIS FILTER

        0702C 0000      (05738)          DATA PITSU$SA            ;APU START ADDRESS
        0702D 0020      (05739)          DATA PITSU$SZ            ;SIZE
                        (05740) ;
                        (05741) PITSU$: BEGIN APU(PITSYN)
   A00 0702E 00E00000 (05742) PITSU$SA: MOV(IQA,M0) \ NOP              ;(2**-15)
   A01 07030 00EC0000 (05743)          MOV(IQA,M4) \ NOP              ;PITCH (FLT PT)
   A02 07032 04000000 (05744)          MUL(M0,M4) \ NOP               ;FIX THE PITCH
   A03 07034 00D00000 (05745)          MOV(P,A0) \ NOP
   A04 07036 3A000000 (05746)          ALIGN(A0) \ NOP
   A05 07038 009C0000 (05747)          MOV(R,OQ) \ NOP                ;OUTPUT FIXED POINT PITCH
                      (05748) ;HERE WE WAIT ON OQE, PROCEED APS, READ/WRITE THE INSTRUCTION, WAIT
                      (05749) ;  AND PROCEED THE APS AGAIN, AND THEN GET DOWN TO THE COMPUTATION.
   A06 0703A 90100006 (05750) 01:      JUMPC(01,OQE)
   A07 0703C 00000000 (05751)          NOP                      ;STORER SAYS WE NEED JUST 1 NOP HERE
   A08 0703E 20372037 (05752)          CLEAR(WI)                ;NOW LET THE APS PROCEED
   A09 07040 00FC0000 (05753)          MOV(IQA,OQ) \ NOP        ;WRITE THE WHOLE INSTR. INTO THE APS
   A0A 07042 901B000A (05754) 02:      JUMPC(02,OQE)            ;WAIT AGAIN
   A0B 07044 00000000 (05755)          NOP
   A0C 07046 20372037 (05756)          CLEAR(WI)                ;AND LET APS PROCEED AGAIN
   A0D 07048 00EC00EE (05757)          MOV(IQA,M4) \ MOV(IQA,M6)      ;FETCH V(0), V(1), V(2)
   A0E 0704A 00ED00EC (05758)          MOV(IQA,M5) \ MOV(IQA,M4)
   A0F 0704C 00EE00ED (05759)          MOV(IQA,M6) \ MOV(IQA,M5)
                      (05760) ;MAIN LOOP
   A10 0704E 00E000E0 (05761) 03:      MOV(IQA,M0)                    ;INPUT Y(N-M+1)
   A11 07050 04118411 (05762)          MOV(A1),MUL(M0,M4)             ;STORE 3RD PROD, START 1ST
   A12 07052 49104910 (05763)          MOV(A0),SUB(A0,A1)             ;STORE 2ND SUM, START 3RD
   A13 07054 00F20000 (05764)          MOV(IQA,A2) \ NOP              ;INPUT U(N) \ U(N+1)
   A14 07056 000000F2 (05765)          NOP \ MOV(IQA,A2)
   A15 07058 009C009C (05766)          MOV(R,OQ)                      ;OUTPUT Y(N-2) \ Y(N-1)
   A16 0705A 00E900E9 (05767)          MOV(IQA,M1)                    ;INPUT Y(N-M)
   A17 0705C 04018401 (05768)          MOV(A1),MUL(M1,M5)             ;STORE 1ST PROD, START 2ND
   A18 0705E 49404940 (05769)          SUB(A2,A1)                     ;START 1ST SUM
   A19 07060 00EA0000 (05770)          MOV(IQA,M2) \ NOP              ;INPUT Y(N-M-1) \ Y(N-M+2)
   A1A 07062 000000EA (05771)          NOP \ MOV(IQA,M2)
   A1B 07064 05510551 (05772)          MOV(A1),MUL(M2,M6)             ;STORE 2ND PROD, START 3RD
   A1C 07066 49104910 (05773)          MOV(A0),SUB(A0,A1)             ;STORE 1ST SUM, START 2ND
   A1D 07068 901C0010 (05774)          JUMPC(03,EO)                   ;LOOP UNLESS OUTPUT COMPLETE
   A1E 0706A 20322032 (05775)          CLEAR(RA)
   A1F 0706C 10000000 (05776)          JUMP(0)
             00000020 (05777) PITSU$SZ=0A-PITSU$SA
        0706E         (05778)          END
```

```
                        (05779) * APS3-PITSYN(Y,A,U,V)       APS PROGRAM FOR PITSYN
                        (05780) ;
                        (05781) ;  J. WOLF  7/1/80
                        (05782) ;
                        (05783) ;THIS APS PROGRAM MODIFIES ONE INSTRUCTION IN ITSELF BY:
                        (05784) ;  0. READING PITCH FROM SCALAR A, AND FIXING IT
                        (05785) ;  1. WRITING IT INTO THE RN OF THE BUS-1 COPY OF THE INSTRUCTION
                        (05786) ;  2. WAITING ON OQE TO BE SURE IT'S THERE (PLUS 1 NOP THAT STORER
                        (05787) ;     SAYS IS NECESSARY)
                        (05788) ;  3. READING THE MODIFIED APS INSTRUCTION (FULLWORD) FROM BUS 1
                        (05789) ;  4. WRITING IT TO APSMEM IN PSEUDOMEMORY
                        (05790) ;  5. WAITING FOR OQE AGAIN BEFORE LETTING THE APS INPUT PROGRAM
                        (05791) ;     PROCEED.
                        (05792) ;
                        (05793) ;HEADER BLOCK
                        (05794) ;
                        (05795)          EVEN
    0706E 000070D6      (05796)          ADDR PITSS$I          ;PTR TO CONSTRUCTED INSTRUCTION BLOCK
    07070 0000707C      (05797)          ADDR PITSSI+2*PITSS$S ;PTR TO SCALAR BLOCK
    07072 0001          (05798)          DATA 1                ;NUMBER OF SCALARS
    07073 0070          (05799)          DATA PITSS$Z          ;MODULE SIZE
    07074 000070C0      (05800)          ADDR PITSS$A          ;PTR TO CHAIN ANCHOR
                        (05801)          EVEN
                        (05802) ;
                        (05803) ;INPUT PROGRAM
                        (05804) ;  BR0: V(X), THEN V(N-M+X)
                        (05805) ;  BR1: U(N)
                        (05806) ;  BR2: LOOP COUNTER, STARTS AT UBS-1
                        (05807) ;
                        (05808) PITSSI: BEGIN APS(PITSS)
A00 07076 00202040      (05809)          JSN(PITSSO,P2)        ;SET AND START OUTPUT PROGRAM
A01 07078 02300030      (05810)          SET(RO)
A02 0707A 04C203CE      (05811)          LOAD(BR0,SVTSUM1(1),TF)  ;GEN (2**-15) ADR
A03 0707C 06C20000      (05812) PITSS$S: LOAD(BR0,M$$(1),TF)   ;GEN SA ADDRESS (PITCH)
A04 0707E 08300037      (05813)          SET(WI)               ;WAIT FOR APU TO SIGNAL WRITE COMPLETE
A05 07080 0A000020      (05814)          NOP(0)
A06 07082 0CC27098      (05815)          LOAD(BR0,PITSS$M(1),TF)  ;LOAD MODIFIED INSTR THRU APU INTO APS MEMORY
A07 07084 0E300037      (05816)          SET(WI)               ;WAIT AGAIN FOR WRITE TO COMPLETE
A08 07086 10000020      (05817)          NOP
A09 07088 12C02000      (05818)          LOAD(BR0,[2],TF)      ;VBA, GEN V(0)
A0A 0708A 14500000      (05819)          LOAD(BR1,M$$)         ;(UNUSED) VLNGTH-1
A0B 0708C 168A0000      (05820)          ADD(BR0,M$$,TF)       ;V INCR, GEN V(1)
A0C 0708E 188AA006      (05821)          ADD(BR0,[10],TF)      ;GEN V(2)
                        (05822) ;NOW LOAD INFO FOR ACCESSING THE Y AND U BUFFERS
A0D 07090 1A400002      (05823)          LOAD(BR0,[0])         ;YBA
A0E 07092 1C600000      (05824)          LOAD(BR2,M$$)         ;(UNUSED) Y LNGTH-1
A0F 07094 1E600000      (05825)          LOAD(BR2,M$$)         ;(UNUSED) Y INCR
A10 07096 200A0100      (05826)          ADD(RR0,216*M$)       ;OFFSET FOR 216 PAST VALUES
                        (05827) ;NEXT INSTRUCTION GETS THE PITCH JAMMED INTO IT!
          00057098      (05828) PIT3S$M=0L
A11 07098 22600000      (05829)          LOAD(BR2,M$$)         ;LOAD PITCH
A12 0709A 2429002D      (05830)          ADDB(BR2,BR2)         ;2*PITCH FOR FULLWORD ADDRESSING
A13 0709C 26090025      (05831)          SUBB(BR0,BR2)         ;POINT TO V(0-M)
```

```
PAGE  130:  <BB4D3><DCA16>BBN16M.NSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APS3-PITSYN(Y,A,U,V)        APS PROGRAM FOR PITSYN

  A14 0709E 2050100F (05032)          LOAD(BR1,C13)           ;UBA
  A15 070A0 2A600000 (05033)          LOAD(BR2,M$$)           ;UBS-1
  A16 070A2 2C120000 (05034)          SUB(BR1,M$$)            ;U INCR
                     (05035) ;MAIN LOOP
  A17 070A4 2E09603A (05036) #4:      ADDL(BR0,W$,TF)         ;Y(N-M+1) (REMEMBER BUFFER Y IS COMPACT)
  A18 070A6 309A900A (05037)          ADD(BR1,C93,TF)         ;U(N)
  A19 070A8 329A9002 (05038)          ADD(BR1,C93,TF)         ;U(N+1)
  A1A 070AA 34090032 (05039)          SUBL(BR0,W$,TF)         ;Y(N-M)
  A1B 070AC 36890032 (05040)          SUBL(BR0,W$,TF)         ;Y(N-M-1)
  A1C 070AE 3809003E (05041)          ADDL(BR0,3*W$,TF)       ;Y(N-M+2)
  A1D 070B0 3A291702 (05042)          SUBL(BR2,2),JUMPP(#4)   ;WE'RE DOING 2 POINTS AT ONCE
  A1E 070B2 3C200031 (05043)          CLEAR(RI)
  A1F 070B4 3E000020 (05044)          NOP(0)
                     (05045) ;
                     (05046) ;OUTPUT PROGRAM
                     (05047) ;  BW0: Y(N)
                     (05048) ;  BW1: LOOP COUNTER (STARTS AT UBS-1)
                     (05049) ;  BW3: SCRATCH
                     (05050) ;
  A20 070B6 40300032 (05051) PITSSO: SET(RA)                  ;START APU
                     (05052) ;GENERATE BUS1 ADR FOR THE INSTRUCTION TO BE MODIFIED (RIGHT HALF)
  A21 070B8 43727899 (05053)          LOAD(BW3,PITSS$N+N$(1),TR)
                     (05054) ;GENERATE PSEUDOMEMORY ADR FOR LOADING THE APS MEMORY
  A22 070BA 44F3FFC0 (05055)          LOAD(BW3,APSMEM(1),TF)
                     (05056) ;NOW GET ON WITH THE "ORDINARY" OUTPUT
  A23 070BC 46F20794 (05057)          LOAD(BW3,DMY$(1),TF)    ;OUTPUT 2 DUMMIES
  A24 070BE 48F20794 (05058)          LOAD(BW3,DMY$(1),TF)
  A25 070C0 4A400018 (05059)          LOAD(BW0,C03)           ;YBA
  A26 070C2 4C700000 (05060)          LOAD(BW3,M$$)           ;(UNUSED) YBS-1
  A27 070C4 4E020000 (05061)          SUB(BW0,M$$)            ;Y INCR
  A28 070C6 500A01B0 (05062)          ADD(BW0,216*W$)         ;POINT TO 1ST OUTPUT SAMPLE-1 IN Y
  A29 070C8 52701000 (05063)          LOAD(BW3,C13)           ;(UNUSED) UBA
  A2A 070CA 54500000 (05064)          LOAD(BW1,M$$)           ;UBS-1
  A2B 070CC 56320000 (05065)          SUB(BW3,M$$)            ;(UNUSED) U INCR
                     (05066) ;OUTPUT LOOP
  A2C 070CE 5801003A (05067) #5:      ADDL(BW0,W$,TF)
  A2D 070D0 5A112CB1 (05068)          SUBL(BW1,1),JUMPP(#5)
  A2E 070D2 5C200030 (05069)          CLEAR(RO)
  A2F 070D4 5E000020 (05070)          NOP(0)
      070D6          (05071)          END
            000070C0 (05072) PITSS$A=#C                       ;ASSIGN VALUE TO CHAIN ANCHOR
      070D6 00000000 (05073) PITSS$I: DATA 0F"0.0"            ;BLOCK FOR CONSTRUCTED INSTRUCTIONS
       ...
            00000070 (05074) PITSS$Z=#L-PITSSI                ;COMPUTE BUS1 MODULE SIZE
```

```
                         ' APU3 - DEAL(Y,A,U,B,V,W)
                         ; KFIELD 9/80
                         ;
                         ; RUNS WITH DEALS$
                         ;
                         ; FUNCTION:
                         ;        DEAL DECODED RECEIVED DATA FROM INPUT BUFFER Y.
                         ;        Y =>    (216) DECODED APC RESIDUAL SAMPLES
                         ;                (  1) DECODED ENERGY
                         ;                (  3) DECODED DELTA GAINS
                         ;                (  1) DECODED PITCH LAG
                         ;                (  3) DECODED PITCH PREDICTOR COEFFS
                         ;                (  6) DECODED SPECTRAL PREDICTOR COEFFS
                         ;                -----
                         ;                (230)
                         ;
                         ;        THE 216 DECODED APC RESIDUAL SAMPLES ARE NOT MOVED FROM THE Y
                         ;        BUFFER.  THEY ARE SUBSEQUENTLY REFERENCED DIRECTLY AS Y(0-215).
                         ;        ALL PARAMETERS (BUFFERS & SCALARS) ARE TREATED AS RL,LWC
                         ;
                         ;        Y(216)      => SA
                         ;        Y(217-219) => U(0-2)
                         ;        Y(220)      => SB
                         ;        Y(221-223) => V(0-2)
                         ;        Y(224-229) => W(0-5)
                         ;
                         ; INPUT SEQUENCE:
                         ;        Y(216), ..., Y(229), [PI]
                         ;
                         ; OUTPUT SEQUENCE:
                         ;        SA, U(0), U(1), U(2),
                         ;        SB, V(0), V(1), V(3),
                         ;        W(0), ..., W(5), [EO]
                         ;        .
                                  EVEN
     070E6 0000                   DATA    DEALU$SA
     070E7 0004                   DATA    DEALU$SZ
                         ;
                         DEALU$  BEGIN   APU(DEALU)
                         DEALU$SA:
 A00 070E8 00FC0000               MOV(IQA,OQ) \ NOP                 ;COPY INPUT TO OUTPUT
 A01 070FA 901C0000               JUMPC(DEALU$SA,EO)                ;LOOP TIL DONE
                         ;
 A02 070FC 20322032               CLEAR(RA)
 A03 070FE 10000000               JUMP(0)
           00000004      DEALU$SZ = $A-DEALU$SA
     070F0                        END
```

```
PAGE  132:  [DBWD]<DCA16>DBWIGN.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APS3 - DEAL(V,A,U,B,V,W)

                          (05922) ' APS3 - DEAL(V,A,U,B,V,W)
                          (05923) ; KFIELD 9/80
                          (05924) ;
                          (05925) ; INPUT SEQUENCE:
                          (05926) ;       V(216), ..., V(229), [FI]
                          (05927) ;
                          (05928) ; OUTPUT SEQUENCE:
                          (05929) ;       SA, U(0), U(1), U(2),
                          (05930) ;       SB, V(0), V(1), V(3),
                          (05931) ;       W(0), ..., W(5), [EO]
                          (05932) ;
                          (05933) ; ALL DATA ITEMS ARE RL,LNG (TF FORMAT)
                          (05934) ;
                          (05935)         EVEN
      070F0 0000713A (05936)         ADDR    DEALS$I               ;PTR TO CONS INS BLOCK
      070F2 00007110 (05937)         ADDR    DEALS$+2*DEALS$S      ;SCALAR BLOCK
      070F4 0002     (05938)         DATA    2                     ;# OF SCALARS
      070F5 0056     (05939)         DATA    DEALS$Z               ;MODULE SIZE
      070F6 00007132 (05940)         ADDR    DEALS$A               ;PTR TO CHAIN ANCHOR
                          (05941) ;
                          (05942) DEALS$: BEGIN   APS(DEALS)
  A00 070F8 00200040 (05943)         JSN(DEALS$O,P2)               ;SET UP & START OUTPUT PGN
  A01 070FA 02300030 (05944)         SET(RD)
                          (05945) ;
                          (05946) ; INPUT PGN
                          (05947) ;
                          (05948) ;
                          (05949) ; GEN V(216-229)
                          (05950) ;
  A02 070FC 04500000 (05951)         LOAD(BR1,[0])                 ;LOAD V(0) ADDR
  A03 070FE 06600000 (05952)         LOAD(BR2,N$$)                 ;DUMMY
  A04 07100 00120000 (05953)         SUB(BR1,N$$)                  ;LOAD V(-1) ADDR
                          (05954) ;
  A05 07102 0A1A01B0 (05955)         ADD(BR1,216*N$)               ;LOAD V(215) ADDR
  A06 07104 0C60000D (05956)         LOAD(BR2,14-1)                ;GEN 14 SAMPLES
                          (05957) ;
  A07 07106 0E9A000A (05958) B1:     ADD(BR1,[0],TF)               ;GEN V ELEM ADDR
  A08 07108 10290781 (05959)         SUBL(BR2,1),JUMPP(B1)
                          (05960) ;
  A09 0710A 12200031 (05961)         CLEAR(RI)                     ;DONE WITH INPUT
  A0A 0710C 14000020 (05962)         NOP(0)
                          (05963)         EJECT
```

```
PAGE  133:  [PBMD]<DCA16>DBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
             APS3 - DEAL(T,A,U,B,V,W)

                     (05964) ;
                     (05965) ; OUTPUT PGM
                     (05966) ;
                     (05967) DEALS$0:
 A0B 0710E 163F0032 (05968)         SET(RA)                        ;START APU
                     (05969) DEALS$S:
 A0C 07110 10C20000 (05970)         LOAD(BW0,M$$(1),TF)            ;GEN SA ADDR
 A0D 07112 1A420000 (05971)         LOAD(BW0,M$$(1))               ;LOAD SB ADDR
                     (05972) ;
                     (05973) ; GEN U(0-2)
                     (05974) ;
 A0E 07114 1C0010EE (05975)         LOAD(BW1,[1],TF)               ;GEN U(0)
 A0F 07116 1E600000 (05976)         LOAD(BW2,M$$)                  ;DUMMY
 A10 07118 20600000 (05977)         LOAD(BW2,M$$)                  ;DUMMY
                     (05978) ;
 A11 0711A 229A9006 (05979)         ADD(BW1,[9],TF)                ;GEN U(1)
 A12 0711C 249A9002 (05980)         ADD(BW1,[9],TF)                ;GEN U(2)
                     (05981) ;
                     (05982) ; GEN SB
                     (05983) ;
 A13 0711E 26010010 (05984)         MOVB(BW0,BW0,TF)               ;GEN SB
                     (05985) ;
                     (05986) ; GEN V(0-2)
                     (05987) ;
 A14 07120 28002004 (05988)         LOAD(BW1,[2],TF)               ;GEN V(0)
 A15 07122 2A600000 (05989)         LOAD(BW2,M$$)                  ;DUMMY
 A16 07124 2C600000 (05990)         LOAD(BW2,M$$)                  ;DUMMY
                     (05991) ;
 A17 07126 2E911006 (05992)         ADD(BW1,[10],TF)               ;GEN V(1)
 A18 07128 309AA002 (05993)         ADD(BW1,[10],TF)               ;GEN V(2)
                     (05994) ;
                     (05995) ; GEN W(0-5)
                     (05996) ;
 A19 0712A 32503002 (05997)         LOAD(BW1,[3])                  ;LOAD W(0) ADDR
 A1A 0712C 34600000 (05998)         LOAD(BW2,M$$)                  ;DUMMY
 A1B 0712E 36120000 (05999)         SUB(BW1,M$$)                   ;LOAD W(-1) ADDR
                     (06000) ;
 A1C 07130 38600005 (06001)         LOAD(BW2,6-1)                  ;GEN 6 SAMPLES
 A1D 07132 3A918000 (06002) 02:     ADD(BW1,[11],TF)               ;GEN W ELEMENT ADDR
 A1E 07134 3C211001 (06003)         SUBL(BW2,1),JUMPP(02)
                     (06004) ;
 A1F 07136 3E200030 (06005)         CLEAR(RO)                      ;DONE WITH OUTPUT
 A20 07138 40000020 (06006)         NOP(0)
                     (06007) ;
          00007132 (06008) DEALS$A=0C                             ;CHAIN ANCHOR
                     (06009) ;
     0713A          (06010)         END
                     (06011) ;
     0713A 00000000 (06012) DEALS$T DATA    10F'0.0'
      ...
                     (06013) ;
          00000056 (06014) DEALS$Z=0L-DEALS$                      ;MODULE SIZE
```

```
                       * APU3-DCORZ DISCRETE CORRELATION APU PROGRAM
                       *      BINDS TO APS3-0211A
                       *
                       *      EQ:  Y(J)=SUM[U(K+J)*V(K)] FOR K=0,1,2,..M-1) FOR J=0,1,2,..N-1
                       *
                       *      WHERE M=SIZE OF REFERENCE, AND N=SIZE OF OUTPUT
                       *
                       *      LEADING AND TRAILING ZEROS ARE AUTOMATICALLY APPLIED BY THE APS PROGRAM
                       *
                       *      ALGORITHM CALCULATES FOUR RUNNING SUMS FOR FOUR SUCCESSIVE POSITIONS
                       *      BETWEEN THE REFERENCE FUNCTION AND THE INPUT, AS FOLLOWS:
                       *
                       *      S1=U(K)*V(K)+U(K+1)*V(K+1)+U(K+2)*V(K+2)+U(K+3)*V(K+3)+...
                       *      S2=U(K)*V(K-1)+U(K+1)*V(K)+U(K+2)*V(K+1)+U(K+3)*V(K+2)+...
                       *      S3=U(K)*V(K-2)+U(K+1)*V(K-1)+U(K+2)*V(K)+U(K+3)*V(K+1)+...
                       *      S4=U(K)*V(K-3)+U(K+1)*V(K-2)+U(K+2)*V(K-1)+U(K+3)*V(K)+...
                       *
                       *      REGISTER ASSIGNMENTS ARE AS FOLLOWS:
                       *
                       *          INPUT-K   K+1      K+2      K+3
                       *       M0:  U(K)     ------   U(K+2)   ------
                       *       M1:  ------   U(K+1)   ------   U(K+3)
                       *       M4:  V(K)     ------   ------   ------
                       *       M5:  V(K-3)   V(K+1)   ------   ------
                       *       M6:  V(K-2)   ------   V(K+2)   ------
                       *       M7:  V(K-1)   ------   ------   V(K+3)
                       *
                       *       A1=S1(K), A2=S2(K-1), A3=S3(K-2), A4=S4(K-3)
                       *
                       *
                       *
          00000003            $M=3
                              EVEN                       START ON FULL WORD BOUNDARY
  0714E 0000                  DATA   DCOR$SA             START ADDRESS
  0714F 0005                  DATA   DCOR$SZ             MODULE SIZE
                       *
                       DCOR$ BEGIN APU(DCORZ)
                       *
A00 07150 00110012     DCOR$SA MOV(ZERO,A1)\MOV(ZERO,A2)          ZERO SUMS S1,S2
A01 07152 00130014            MOV(ZERO,A3)\MOV(ZERO,A4)           ZERO SUMS S3,S4
A02 07154 00090009            MOV(ZERO,M1)                        ZERO U(-1)
A03 07156 000D000D            MOV(ZERO,M5)                        ZERO V(-3)
A04 07158 000E000E            MOV(ZERO,M6)                        ZERO V(-2)
A05 0715A 000F000F            MOV(ZERO,M7)                        ZERO V(-1)
A06 0715C 04E004C0            MUL(M1,M7)\MUL(M1,M6)               PS1(-1);PS2(-2)
A07 0715E 02600200            R(A3)\R(A4)                         S3(-3);S4(-4)
A08 07160 911C0007            JUMPS(DCEND,EQ)
                       *
                       *      INNER LOOP CALCULATES FOUR SUMS AT A TIME
                       *      FOR FOUR PAIRS OF U,V PER PASS FOR K=0,4,8,...M-1
                       *
                       *
A09 07162 00E000E0     DCOR0 MOV(IQA,M0)                                M0=U(K)
```

```
A0A 07164 04550435 (06068)        MOV(A5),MUL(M0,M6)\MOV(A5),MUL(M0,M5)  A5=PS1(K-1),PS3(K-2);A5=PS2(K-2),PS
A0B 07166 45334554 (06069)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  A3=S3(K-3),S1(K-1);A4=S4(K-4),S2(K-
A0C 07168 00EC00EC (06070)        MOV(IQA,M4)                            M4=V(K)
A0D 0716A 04160476 (06071)        MOV(A6),MUL(M0,M4)\MOV(A6),MUL(M0,M7)  A6=PS3(K-2),PS1(K);A6=PS4(K-3),PS2(
A0E 0716C 46714692 (06072)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  A1=S1(K-1),S3(K-2);A2=S2(K-2),S4(K-
A0F 0716E 91160026 (06073)        JUMPS(DCL0,FWI)                        EXIT IF K=M-1 (LAST REF SAMPLE INDE
                   (06074) *
A10 07170 00E900E9 (06075) DCOR1  MOV(IQA,M1)                            M1=U(K+1)
A11 07172 04F504D5 (06076)        MOV(A5),MUL(M1,M7)\MOV(A5),MUL(M1,M6)  A5=PS1(K),PS3(K-1);A5=PS2(K-1),PS4(
A12 07174 45334554 (06077)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  A3=S3(K-2),S1(K);A4=S4(K-3),S2(K-1)
A13 07176 00ED00ED (06078)        MOV(IQA,M5)                            M5=V(K+1)
A14 07178 04D60496 (06079)        MOV(A6),MUL(M1,M5)\MOV(A6),MUL(M1,M4)  A6=PS3(K-1),PS1(K+1);A6=PS4(K-2),PS
A15 0717A 46714692 (06080)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  A1=S1(K),S3(K-1);A2=S2(K-1),S4(K-2)
A16 0717C 91160030 (06081)        JUMPS(DCL1,FWI)                        EXIT IF K+1=M-1 (LAST REF SAMP INDE
                   (06082) *
A17 0717E 00E800E8 (06083) DCOR2  MOV(IQA,M0)                            M0=U(K+2)
A18 07180 04150475 (06084)        MOV(A5),MUL(M0,M4)\MOV(A5),MUL(M0,M7)  A5=PS1(K+1),PS3(K);A5=PS2(K),PS4(K-
A19 07182 45334554 (06085)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  A3=S3(K-1),S1(K+1);A4=S4(K-2),S2(K)
A1A 07184 00EE00EE (06086)        MOV(IQA,M6)                            M6=V(K+2)
A1B 07186 04560436 (06087)        MOV(A6),MUL(M0,M6)\MOV(A6),MUL(M0,M5)  A6=PS3(K),PS1(K+2);A6=PS4(K-1),PS2(
A1C 07188 46714692 (06088)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  A1=S1(K+1),S3(K);A2=S2(K),S4(K-1)
A1D 0718A 91160054 (06089)        JUMPS(DCL2,FWI)                        EXIT IF K+2=M-1 (LAST REF SAMP INDE
                   (06090) *
A1E 0718C 00E900E9 (06091) DCOR3  MOV(IQA,M1)                            M1=U(K+3)
A1F 0718E 04D50495 (06092)        MOV(A5),MUL(M1,M5)\MOV(A5),MUL(M1,M4)  A5=PS1(K+2),PS3(K+1);A5=PS2(K+1),PS
A20 07190 45334554 (06093)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  A3=S3(K),S1(K+2);A4=S4(K-1),S2(K+1)
A21 07192 00EF00EF (06094)        MOV(IQA,M7)                            M7=V(K+3)
A22 07194 04F604D6 (06095)        MOV(A6),MUL(M1,M7)\MOV(A6),MUL(M1,M6)  A6=PS3(K+1),PS1(K+3);A6=PS4(K),PS2(
A23 07196 46714692 (06096)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  A1=S1(K+2),S3(K+1);A2=S2(K+1),S4(K)
A24 07198 91160068 (06097)        JUMPS(DCL3,FWI)                        EXIT IF K+3=M-1 (LAST REF SAMP INDE
                   (06098) *
A25 0719A 10000089 (06099)        JUMP(DCOR0)
                   (06100) *
                   (06101) *      ENTRY FOR K=M-1 CLEAN-UP
A26 0719C 20372037 (06102) DCL0:  CLEAR(WI)
A27 0719E 00E900E9 (06103)        MOV(IQA,M1)                            M1=U(M)
A28 071A0 04F504D5 (06104)        MOV(A5),MUL(M1,M7)\MOV(A5),MUL(M1,M6)  A5=PS1(M-1),PS3(M-2);A5=PS2(M-2),PS
A29 071A2 45334554 (06105)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  A3=S3(M-3),S1(M-1);A4=S4(M-4),S2(M-
A2A 071A4 04D60496 (06106)        MOV(A6),MUL(M1,M5)\MOV(A6),MUL(M1,M4)  A6=PS3(M-2),PS1(M);A6=PS4(M-3),PS2(
A2B 071A6 467C4692 (06107)        MOV(DQ),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  OUT=S1(M-1),S3(M-2);A2=S2(M-2),S4(M
A2C 071A8 00110011 (06108)        MOV(ZERO,A1)                           CLEAR S1 FOR NEW SUM
                   (06109) *
A2D 071AA 00E800E8 (06110)        MOV(IQA,M0)                            M0=U(M+1)
A2E 071AC 04150475 (06111)        MOV(A5),MUL(M0,M4)\MOV(A5),MUL(M0,M7)  A5=PS1(M),PS3(M-1);A5=PS2(M-1),PS4(
A2F 071AE 45334554 (06112)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  A3=S3(M-2),S1(M);A4=S4(M-3),S2(M-1)
A30 071B0 04560436 (06113)        MOV(A6),MUL(M0,M6)\MOV(A6),MUL(M0,M5)  A6=PS3(M-1),PS1(M+1);A6=PS4(M-2),PS
A31 071B2 911C0082 (06114)        JUMPS(DCEND,EO)                        JUMP TO END IF NO MORE OUTPUT
A32 071B4 4671469C (06115)        MOV(A1),ADD(A3,A6)\MOV(DQ),ADD(A4,A6)  A1=S1(M),S3(M-1);OUT=S2(M-1),S4(M-2
A33 071B6 00120012 (06116)        MOV(ZERO,A2)                           CLEAR S2 FOR NEW SUM
                   (06117) *
A34 071B8 00E900E9 (06118)        MOV(IQA,M1)                            M1=U(M+2)
A35 071BA 04D50495 (06119)        MOV(A5),MUL(M1,M5)\MOV(A5),MUL(M1,M4)  A5=PS1(M+1),PS3(M);A5=PS2(M),PS4(M-
A36 071BC 911C0082 (06120)        JUMPS(DCEND,EO)                        JUMP TO END IF NO MORE OUTPUT
```

```
PAGE  136:  [BBND]<DCA16>BBN16N.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            APU3-DCDRZ DISCRETE CORRELATION APU PROGRAM

 A37 071BE 453C4554 (06121)        MOV(DQ),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  OUT=S3(N-1),S1(N+1);A4=S4(N-2),S2(N
 A38 071C0 00130013 (06122)        MOV(ZERO,A3)                           CLEAR S3 FOR NEW SUM
 A39 071C2 04F604D6 (06123)        MOV(A6),MUL(M1,M7)\MOV(A6),MUL(M1,M6)  A6=PS3(N),PS1(N+2);A6=PS4(N-1),PS2(
 A3A 071C4 46714692 (06124)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  A1=S1(N+1),S3(N);A2=S2(N),SS4(N-1)
 A3B 071C6 0000009C (06125)        NOP\MOV(R,DQ)
 A3C 071C8 10000000 (06126)        JUMP(DCOR$SA)                          CONTINUE TO K+1 PHASE
                    (06127) *
                    (06128) *       ENTRY FOR  K+1 CLEAN-UP
 A3D 071CA 20372037 (06129) DCL1:  CLEAR(WI)
 A3E 071CC 00E000E0 (06130)        MOV(IQA,M0)                            M0=U(N)
 A3F 071CE 04150475 (06131)        MOV(A5),MUL(M0,M4)\MOV(A5),MUL(M0,M7)
 A40 071D0 45334554 (06132)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)
 A41 071D2 04560436 (06133)        MOV(A6),MUL(M0,M6)\MOV(A6),MUL(M0,M5)
 A42 071D4 467C4692 (06134)        MOV(DQ),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  OUT=S1(N-1)
 A43 071D6 00110011 (06135)        MOV(ZERO,A1)                           CLEAR S1 FOR NEW SUM
                    (06136) *
 A44 071D8 00E900E9 (06137)        MOV(IQA,M1)                            M1=U(N+1)
 A45 071DA 04B50495 (06138)        MOV(A5),MUL(M1,M5)\MOV(A5),MUL(M1,M4)
 A46 071DC 911C0002 (06139)        JUMPS(DCEND,EO)                        JUMP OUT IF S1(N-1) IS LAST OUTPUT
 A47 071DE 45334554 (06140)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)
 A48 071E0 04F604D6 (06141)        MOV(A6),MUL(M1,M7)\MOV(A6),MUL(M1,M6)
 A49 071E2 4671469C (06142)        MOV(A1),ADD(A3,A6)\MOV(DQ),ADD(A4,A6)  ;OUT=S2(N-1)
 A4A 071E4 00120012 (06143)        MOV(ZERO,A2)                           CLEAR S2 FOR NEW SUM
                    (06144) *
 A4B 071E6 00E000E0 (06145)        MOV(IQA,M0)                            M0=U(N+2)
 A4C 071E8 04550435 (06146)        MOV(A5),MUL(M0,M6)\MOV(A5),MUL(M0,M5)
 A4D 071EA 911C0002 (06147)        JUMPS(DCEND,EO)                        JUMP OUT IF S2(N-1) IS LAST OUTPUT
 A4E 071EC 453C4554 (06148)        MOV(DQ),ADD(A1,A5)\MOV(A4),ADD(A2,A5)  OUT=S3(N-1)
 A4F 071EE 00130013 (06149)        MOV(ZERO,A3)                           CLEAR A3 FOR NEW S3 SUM
 A50 071F0 04160476 (06150)        MOV(A6),MUL(M0,M4)\MOV(A6),MUL(M0,M7)
 A51 071F2 46714692 (06151)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)
 A52 071F4 0000009C (06152)        NOP\MOV(R,DQ)
 A53 071F6 10000000 (06153)        JUMP(DCOR$SA)
                    (06154) *
                    (06155) *      ENTRY FOR K+2 CLEAN-UP
 A54 071F8 20372037 (06156) DCL2:  CLEAR(WI)
 A55 071FA 00E900E9 (06157)        MOV(IQA,M1)                            M1=U(N)
 A56 071FC 04B50495 (06158)        MOV(A5),MUL(M1,M5)\MOV(A5),MUL(M1,M4)
 A57 071FE 45334554 (06159)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)
 A58 07200 04F604D6 (06160)        MOV(A6),MUL(M1,M7)\MOV(A6),MUL(M1,M6)
 A59 07202 467C4692 (06161)        MOV(DQ),ADD(A3,A6)\MOV(A2),ADD(A4,A6)  OUT=S1(N-1)
 A5A 07204 00110011 (06162)        MOV(ZERO,A1)                           CLEAR A1 FOR NEW S1 SUM
                    (06163) *
 A5B 07206 00E000E0 (06164)        MOV(IQA,M0)                            M0=U(N+1)
 A5C 07208 04550435 (06165)        MOV(A5),MUL(M0,M6)\MOV(A5),MUL(M0,M5)
 A5D 0720A 911C0002 (06166)        JUMPS(DCEND,EO)                        JUMP OUT IF S1(N-1) FINAL OUTPUT
 A5E 0720C 45334554 (06167)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)
 A5F 0720E 04160476 (06168)        MOV(A6),MUL(M0,M4)\MOV(A6),MUL(M0,M7)
 A60 07210 4671469C (06169)        MOV(A1),ADD(A3,A6)\MOV(DQ),ADD(A4,A6)  OUT=S2(N-1)
 A61 07212 00120012 (06170)        MOV(ZERO,A2)                           CLEAR A2 FOR NEW S2 SUM
                    (06171) *
 A62 07214 00E900E9 (06172)        MOV(IQA,M1)                            M1=U(N+2)
 A63 07216 04F504D5 (06173)        MOV(A5),MUL(M1,M7)\MOV(A5),MUL(M1,M6)
```

```
A64 07218 911C0002 (06174)        JUMPS(DCEND,EO)                                 JUMP OUT IF S2(N-1) FINAL OUTPUT
A65 0721A 453C4554 (06175)        MOV(DQ),ADD(A1,A5)\MOV(A4),ADD(A2,A5)          OUT=S3(N-1)
A66 0721C 00130813 (06176)        MOV(ZERO,A3)
A67 0721E 84068496 (06177)        MOV(A6),MUL(M1,M5)\MOV(A6),MUL(M1,M4)
A68 07220 46714692 (06178)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)
A69 07222 0000009C (06179)        NOP\MOV(R,DQ)
A6A 07224 10000000 (06180)        JUMP(DCOR$SA)
                   (06181) *
                   (06182) *      PHASE K+3 CLEAN-UP
A6B 07226 20372037 (06183) DCL3:  CLEAR(MI)
A6C 07228 00E000E0 (06184)        MOV(IQA,M0)                                     M0=U(N)
A6D 0722A 84550435 (06185)        MOV(A5),MUL(M0,M6)\MOV(A5),MUL(M0,M5)
A6E 0722C 45334554 (06186)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)
A6F 0722E 84168476 (06187)        MOV(A6),MUL(M0,M4)\MOV(A6),MUL(M0,M7)
A70 07230 467C4692 (06188)        MOV(DQ),ADD(A3,A6)\MOV(A2),ADD(A4,A6)          OUT=S1(N-1)
A71 07232 00110811 (06189)        MOV(ZERO,A1)
                   (06190) *
A72 07234 00E908E9 (06191)        MOV(IQA,M1)                                     M1=U(N+1)
A73 07236 84F584D5 (06192)        MOV(A5),MUL(M1,M7)\MOV(A5),MUL(M1,M6)
A74 07238 911C0002 (06193)        JUMPS(DCEND,EO)                                 JUMP OUT IF S1(N-1) FINAL OUTPUT
A75 0723A 45334554 (06194)        MOV(A3),ADD(A1,A5)\MOV(A4),ADD(A2,A5)
A76 0723C 84068496 (06195)        MOV(A6),MUL(M1,M5)\MOV(A6),MUL(M1,M4)
A77 0723E 4671469C (06196)        MOV(A1),ADD(A3,A6)\MOV(DQ),ADD(A4,A6)          ;OUT=S2(N-1)
A78 07240 00120812 (06197)        MOV(ZERO,A2)
                   (06198) *
A79 07242 00E000E0 (06199)        MOV(IQA,M0)                                     M0=U(N+2)
A7A 07244 84150475 (06200)        MOV(A5),MUL(M0,M4)\MOV(A5),MUL(M0,M7)
A7B 07246 911C0002 (06201)        JUMPS(DCEND,EO)                                 JUMP OUT IF S2(N-1) FINAL OUTPUT
A7C 07248 453C4554 (06202)        MOV(DQ),ADD(A1,A5)\MOV(A4),ADD(A2,A5)          OUT=S3(N-1)
A7D 0724A 00130813 (06203)        MOV(ZERO,A3)
A7E 0724C 84568436 (06204)        MOV(A6),MUL(M0,M6)\MOV(A6),MUL(M0,M5)
A7F 0724E 46714692 (06205)        MOV(A1),ADD(A3,A6)\MOV(A2),ADD(A4,A6)
A80 07250 0000009C (06206)        NOP\MOV(R,DQ)
A81 07252 10000000 (06207)        JUMP(DCOR$SA)
                   (06208) *
                   (06209) *
                   (06210) *
A82 07254 00000000 (06211)  DCEND: NOP
A83 07256 20322032 (06212)        CLEAR(RA)
A84 07258 10000000 (06213)        JUMP(DCOR$SA)
          00000085 (06214)        DCOR$SZ=$A-DCOR$SA
    0725A          (06215)        END
                   (06216) ;
                   (06217) ;
                   (06218) ;
                   (06219) ;
```

```
                  (06220) * APS3-ADCOR APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL
                  (06221) ;BUFFER ATTRIBUTES FOR INITIAL TESTING
         00000000 (06222) UBASE=0
         00000032 (06223) VBASE=50
         00000064 (06224) YBASE=100
                  (06225) ;
         0000000C (06226) UZ=12                                  ;UZ=USIZE
         00000005 (06227) VZ=5                                   ;VZ=VSIZE
         0000000F (06228) YZ=15                                  ;YZ=YSIZE
         00000002 (06229) NZ=2                                   ;# OF LEADING ZEROS
         00000002 (06230) UI=2                                   ;UI = U SAMPLE INCREMENT
         00000002 (06231) VI=2                                   ;VI = V SAMPLE INCREMENT
         00000002 (06232) YI=2                                   ;YI = Y SAMPLE INCREMENT
                  (06233) ;
                  (06234) ; BINDING DONE BY SBDCORZ
                  (06235) ;
                  (06236) ;
                  (06237) ;APS INPUT SEQUENCE (FOR UZ=12,VZ=5,YZ=15,NZ=2):
                  (06238) ;Z1,V(0),Z2,V(1),U(0),V(2),U(1),V(3),U(2),V(4);U(3),U(4),U(5)
                  (06239) ;U(2),V(0),U(3),V(1),U(4),V(2),U(5),V(3),U(6),V(4);U(7),U(8),U(9)
                  (06240) ;U(6),V(0),U(7),V(1),U(8),V(2),U(9),V(3),U(10),V(4);U(11),Z1,Z2
                  (06241) ;U(10),V(0),U(11),V(1),Z3,V(2),Z4,V(3),Z5,V(4);Z6,(Z7),(Z8),...
                  (06242) ;
                  (06243)      EVEN
    0725A 00000000 (06244)      ADDR $0                          ;PTR TO CONSTRUCTED INST. BLOCK
    0725C 0000726C (06245)      ADDR ADC$+2*ADC$S                ;PTR TO SCALAR BLOCK
    0725E 0000     (06246)      DATA $0                          ;NUMBER OF SCALARS
    0725F 009C     (06247)      DATA ADC$Z                       ;MODULE SIZE
    07260 00000000 (06248)      ADDR ADC$A                       ;PTR TO CHAIN ANCHOR
                  (06249)      EVEN
                  (06250) ;APS OUTPUT SEQUENCE:
                  (06251) ;Y(0),Y(1),....Y(YZ-1),EO WHERE YZ DETERMINES THE NUMBER OF SAMPLES PROCESSED
                  (06252) ;
                  (06253) ADC$:     BEGIN APS(ADCOR)
A00 07262 00204240 (06254)          JSN(ADC$15,P2)
A01 07264 02300030 (06255)          SET(RO)
                  (06256) ;
A02 07266 04640032 (06257) ADC$1:   LOAD(BR2,VBASE(2))           ;LOAD V-BUFFER TRIPLET
A03 07268 06700004 (06258) ADC$1A:  LOAD(BR3,VZ-1)               ;BR3 = VZ-1 = VSIZE-1
A04 0726A 08220002 (06259) ADC$1B:  SUB(BR2,VI)                  ;BR2 = VI = V SAMPLE INCREMENT
                  (06260) ;
A05 0726C 0A500002 (06261) ADC$S:   LOAD(BR1,NZ)                 ;GET # LEADING ZEROS
                  (06262) ; NEXT LINE ADDED BY KFIELD 7/80
A06 0726E 0C390039 (06263)         ADDL(BR3,1)
A07 07270 0E190CF1 (06264)          SUBL(BR1,1),JUMPN(ADC$2)      ;EXIT TO INITIAL ENTRY IF NO ZEROS REQ
                  (06265) ;
A08 07272 10C207BA (06266) $1:      LOAD(BR0,ZERO(1),TF)         ;GEN ZERO LOC
A09 07274 12AA0002 (06267) ADC$1B2: ADD(BR2,VI,TF)               ;GEN V(K)
A0A 07276 14190CF1 (06268)          SUBL(BR1,1),JUMPN(ADC$2)      ;EXIT IF LAST ZERO GENERATED
A0B 07278 163900B1 (06269)          SUBL(BR3,1),JUMPP($1)         ;REPEAT IF MORE V(K)
                  (06270) ;
                  (06271) ;INITIAL ENTRY FOR DATA IN U-BUFFER
A0C 0727A 18440000 (06272) ADC$2:   LOAD(BR0,UBASE(2))           ;U-BUFFER TRIPLET
```

APS3-ADCOR APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL

```
A0D 0727C 1A500000 (06273) ADC$2A: LOAD(BR1,UZ-1)             ;U = UZ-1 = USIZE-1
A0E 0727E 1C020002 (06274) ADC$2B: SUB(BR0,UI)                ;BR0 = UI = U SAMPLE INCREMENT
A0F 07280 1E190039 (06275)         ADDL(BR1,1)
A10 07282 201A0002 (06276) ADC$2C:  ADD(BR1,NZ)                ;BR1=USIZE+NZ
                   (06277) ;
                   (06278) ;
A11 07284 22390031 (06279)         SUBL(BR3,1)                ;DECR V-COUNT
A12 07286 24020004 (06280) ADC$3:  SUB(BR0,NZ*UI)
A13 07288 26010011 (06281)         MOVB(BW0,BR0)              ;SAVE UBASE
A14 0728A 200A0004 (06282) ADC$4:  ADD(BR0,NZ*UI)
A15 0728C 2A110013 (06283)         MOVB(BW1,BR1)              ;SAVE UZ-1
                   (06284) ;
                   (06285) ;PRIMARY LOOP FOR INPUT DATA WITHIN U-BUFFER
A16 0728E 2C1930F1 (06286) 02      SUBL(BR1,1),JUMPN(D211A8)  ;GO TO 2NDARY LOOP IF LAST SAMPLE IN U-BU
A17 07290 2E0A0002 (06287) ADC$5:  ADD(BR0,UI,TF)             ;GEN U(K) ADDR
A18 07292 30AA0002 (06288) ADC$6:  ADD(BR2,VI,TF)             ;GEN V(K) ADDR
A19 07294 32391601 (06289)         SUBL(BR3,1),JUMPP(02)      ;LAST SAMPLE IN V REF TEST
A1A 07296 343E0037 (06290)         SET(WI)
A1B 07298 36000020 (06291)         NOP(0)
                   (06292) ;GENERATE U(N),U(N+1),U(N+2)
A1C 0729A 38192BF1 (06293)         SUBL(BR1,1),JUMPN(D211A5)  ;EXIT IF NEXT U(K) BEYOND U-BUFFER
A1D 0729C 3A0A0002 (06294) ADC$7:  ADD(BR0,UI,TF)             ;GEN U(N)
A1E 0729E 3C192CF1 (06295)         SUBL(BR1,1),JUMPN(D211A6)  ;EXIT IF NEXT U(K) BEYOND U-BUFFER
A1F 072A0 3E0A0002 (06296) ADC$8:  ADD(BR0,UI,TF)             ;GEN U(N+1)
A20 072A2 40192DF1 (06297)         SUBL(BR1,1),JUMPN(D211A7)  ;EXIT IF NEXT U(K) BEYOND U-BUFFER
A21 072A4 420A0002 (06298) ADC$9:  ADD(BR0,UI,TF)             ;GEN U(N+2)
                   (06299) ;
A22 072A6 44640032 (06300) ADC$10:  LOAD(BR2,VBASE(2))        ;RELOAD V
A23 072A8 46700004 (06301) ADC$10A: LOAD(BR3,VZ-1)
A24 072AA 48220002 (06302) ADC$10B: SUB(BR2,VI)
                   (06303) ;
A25 072AC 4A090010 (06304)         MOVB(BR0,BW0)              ;GET UBASE
A26 072AE 4C0A0000 (06305) ADC$11: ADD(BR0,4*UI)              ;ADVANCE UBASE FOR NEXT GROUP
A27 072B0 4E010011 (06306)         MOVB(BW0,BR0)              ;SAVE IT FOR NEXT TIME
A28 072B2 50110034 (06307)         SUBL(BW1,4)                ;DECREASE USIZE COUNT FOR NEXT GROUP
A29 072B4 52190012 (06308)         MOVB(BR1,BW1)              ;PLACE IT INTO U-COUNTER
A2A 072B6 54001660 (06309)         JUMP(02)
                   (06310) ;
                   (06311) ;ENTRIES FOR U(K) OUTSIDE U-BUFFER, ZEROS NEEDED
A2B 072B8 56C207BA (06312) D211A5: LOAD(BR0,ZERO(1),TF)
A2C 072BA 58C207BA (06313) D211A6: LOAD(BR0,ZERO(1),TF)
A2D 072BC 5AC207BA (06314) D211A7: LOAD(BR0,ZERO(1),TF)
A2E 072BE 5C000020 (06315)         NOP(0)
A2F 072C0 5E002260 (06316)         JUMP(ADC$10)
                   (06317) ;
                   (06318) ;SECONDARY LOOP ENTEERED WHEN LAST U(K) IN UBUFFER USED
A30 072C2 60C207BA (06319) D211A8: LOAD(BR0,ZERO(1),TF)
A31 072C4 62AA0002 (06320) ADC$12: ADD(BR2,VI,TF)
A32 072C6 64393001 (06321)         SUBL(BR3,1),JUMPP(D211A8)
A33 072C8 66300037 (06322)         SET(WI)
A34 072CA 68000020 (06323)         NOP(0)
A35 072CC 6AC207BA (06324)         LOAD(BR0,ZERO(1),TF)
A36 072CE 6CC207BA (06325)         LOAD(BR0,ZERO(1),TF)
```

```
PAGE  140:  [BBND]<DCA16>BWN16M.MSQ.2, 29-Dec-80 14:30:40, Ed: WOLF
                      APS3-ADCOR APS PROGRAM FOR DISCRETE CORRELATION WITH AUTO ZERO FILL

  A37 072D0 6EC207BA (06326)          LOAD(BR0,ZERO(1),TF)
                     (06327) ;
  A38 072D2 70640032 (06328) ADC$13: LOAD(BR2,YBASE(2))
  A39 072D4 72700004 (06329) ADC$13A: LOAD(BR3,YZ-1)
  A3A 072D6 74220002 (06330) ADC$13B: SUB(BR2,YI)
                     (06331) ;
  A3B 072D8 76002260 (06332)          JUMP(ADC$10)
  A3C 072DA 780A0006 (06333) ADC$14: ADD(BR0,3*UI)
  A3D 072DC 7A010011 (06334)          MOVB(BW0,BR0)
  A3E 072DE 7C090010 (06335)          MOVB(BR0,BW0)
  A3F 072E0 7E110033 (06336)          SUBL(BW1,3)
  A40 072E2 80190012 (06337)          MOVB(BR1,BW1)
  A41 072E4 82003060 (06338)          JUMP(D211A0)
                     (06339) ;
                     (06340) ;OUTPUT PROGRAM
  A42 072E6 84300032 (06341) ADC$15:  SET(RA)
  A43 072E8 86640064 (06342) ADC$16:  LOAD(BW2,YBASE(2))
  A44 072EA 8870000E (06343) ADC$16A: LOAD(BW3,YZ-1)
  A45 072EC 8A220002 (06344) ADC$16B: SUB(BW2,YI)
                     (06345) ;
  A46 072EE 8CAA0002 (06346) ADC$17: ADD(BW2,YI,TF)
  A47 072F0 8E3146B1 (06347)          SUBL(BW3,1),JUMPP(ADC$17)
  A48 072F2 90000020 (06348)          NOP(0)
  A49 072F4 92000020 (06349)          NOP(0)
  A4A 072F6 94200030 (06350)          CLEAR(RO)
  A4B 072F8 96000020 (06351)          NOP(0)
  A4C 072FA 98000020 (06352)          NOP(0)
  A4D 072FC 9A000020 (06353)          NOP(0)
            00000000 (06354)          ADC$A=0C                            ;ASSIGN VALUE TO CHAIN
      072FE          (06355)          END
            0000009C (06356)          ADC$Z=0L-ADC$
                     (06357) ;
                     (06358) ;
                     (06359) ;
                     (06360) ;
```

AD-A096 092 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL DCA100-79-C-0037
UNCLASSIFIED BBN-4565-VOL-2 NL

4 OF 5
AD A
096092

```
                   (06361) ' SBDCORZ - SPECIAL BINDING MODULE FOR DCORZ
                   (06362) ;
                   (06363) ; CALLING SEQUENCE
                   (06364) ;  CALL R1,SBDCORZ
                   (06365) ;
                   (06366) ; WHERE  (R1)=POINTER TO FCB NUMBER IN FCB
                   (06367) ;        (R2)=POINTER TO ARRAY FUNCTION DISPATCH INFO
                   (06368) ;        (R3)=ORIGIN FOR PENDING SUPPORT LIST
                   (06369) ;
                   (06370) ; MODIFIED BY KFIELD 7/80
                   (06371) ; USE SAF'S FOR SAMPLE INCREMENTS
                   (06372) ; (TO HANDLE NEGATIVE SI'S).
                   (06373) ;
072FE C63400E8 (06374) SBDCORZ:PUSHMIL R3,AFDT$ORG(R2)          ;STACK APU MODULE...
07300 9071FFFE (06375)         MOVIR R7,-W$
07302 C6B7FFFE (06376)         PUSHMIL R3,0-W$(R3)              ;STACK APU MODULE S.A. AND SIZE
07304 C63400EA (06377)         PUSHMIL R3,AFDT$ORG+W$(R2)       ;STACK APS MODULE BUS ORIGIN
07306 C63400EC (06378)         PUSHMIL R3,AFDT$ORG+2*W$(R2)     ;STACK CSPU MODULE BUS ORIGIN
07308 2771     (06379)         DECR R7,1
07309 0000     (06380)         EVEN
0730A C4B7FFFC (06381)         PUSHMI R3,0-2*W$(R3)             ;APS MODULE SIZE
0730C 9047FFFD (06382)         MOVIR R4,-3*H$(R3)               ;GET SPECIAL SUPPORT SEMAPHORE
0730E 704AFF00 (06383)         MOVWK R4,R4,MSK$LBYT
07310 3C48     (06384)         LRS R4,8
07311 3438     (06385)         PUSHXI R3,R4                     ;STACK SPECIAL SUPPORT SEMAPHORE
07312 4052     (06386)         MOVRR R5,R1                      ;POINT TO FIRST SCALAR ID
07313 2651     (06387)         INCR R5,H$
07314 704A00FF (06388)         MOVWK R4,R5,MSK$RBYT             ;EXTRACT SA ID
07316 3638     (06389)         PUSHXIL R3,R4                    ;STACK SA ID
07317 C6300792 (06390)         PUSHMIL R3,CLR$G0G1              ;RESET 6 FLAGS
07319 2632     (06391)         INCR R3,2
0731A C63007BA (06392)         PUSHMIL R3,ZERO                  ;SET FOR NO PBP
0731C F030024D (06393)         MOVMR R3,AP$BSL                  ;POINT TO START OF SUPPORT LIST
                   (06394) ;
                   (06395) ; SCALAR BINDING
                   (06396) ;
                   (06397) ;
                   (06398) ; NZ BINDING
                   (06399) ; NZ=# OF LEADING ZEROS
                   (06400) ;
0731E F0420001 (06401)         MOVMR R4,H$(R1)                  ;GET SCALAR NZ ID
07320 9A4000FF (06402)         ANDIR R4,MSK$RBYT                ;MASK RIGHT BYTE
07322 9C400502 (06403)         ADDIR R4,ISVT$                   ;R4=R4+ISVT$
07324 6038     (06404)         MOVWR R3,R4                      ;R3=NZ
07325 E030726D (06405)         MOVRM R3,ADC$+W$*ADC$S+H$        ;STORE IN APS MAIN MEMORY
07327 E0307283 (06406)        MOVRM R3,ADC$+W$*ADC$2C+H$
                   (06407) ;
                   (06408) ; UBUFFER BINDING
                   (06409) ; UBASE BINDING
07329 F0620002 (06410)         MOVMR R6,2*H$(R1)                ;GET U ID
0732B 506CFF00 (06411)         MOVKR R6,R6,MSK$LBYT             ;GET PROPER BYTE
0732D 3C67     (06412)         LRS R6,7                         ;CONVERT TO BCT$ INDEX
                   (06413)         EVEN
```

```
07332E C04C0582 (06414)        MOVMRL R4,BCT$BA(R6)             ;GET 17-BIT UBASE ADDR
07330 F070727A (06415)         MOVMR R7,ADC$+W$*ADC$2           ;APS ADR IN MAIN MEMORY
07332 564EFFF0 (06416)         IORKR R4,R7,$FFF0
07334 8440727A (06417)         MOVRML R4,ADC$+W$*ADC$2          ;MODIFIED APS INSTR IN MAIN MEMORY
               (06418) ;
               (06419) ; (UZ-1) BINDING
               (06420) ; (UZ-1) = USIZE - 1 = U BS
07336 F04C0605 (06421)        MOVHR R4,H$+BCT$AD(R6)           ;GET BS = # OF COLS - 1
07338 F0407270 (06422)        MOVRM R4,ADC$+W$*ADC$2A+H$       ;MOVE TO APS MAIN MEMORY
               (06423) ;
               (06424) ; UI BINDING
               (06425) ; UI=U SAMPLE INCREMENT
0733A F05C0604 (06426)        MOVHR R5,BCT$AD(R6)               ;R5=U SI
0733C 407A     (06427)        MOVRR R7,R5                        ;R7=R5=U SI
0733D EF50727E (06428)        SAF R5,ADC$+W$*ADC$20          ;MOVE TO APS MAIN MEMORY
0733F EF507290 (06429)        SAF R5,ADC$+W$*ADC$5           ;MOVE TO APS MAIN MEMORY
07341 EF50729C (06430)        SAF R5,ADC$+W$*ADC$7           ;MOVE TO APS MAIN MEMORY
07343 EF5072A0 (06431)        SAF R5,ADC$+W$*ADC$8           ;MOVE TO APS MAIN MEMORY
07345 EF5072A4 (06432)        SAF R5,ADC$+W$*ADC$9           ;MOVE TO APS MAIN MEMORY
               (06433) ;
               (06434) ; 4*UI BINDING
               (06435) ; 4*UI=4*(U SAMPLE INCREMENT)
07347 3A52     (06436)        LLS R5,2                           ;R5=SI .L. 2=4*UI
07348 EF5072AE (06437)        SAF R5,ADC$+W$*ADC$11          ;MOVE TO APS MAIN MEMORY
               (06438) ;
               (06439) ; NZ*UI BINDING
               (06440) ; (NZ*UI) = (# OF LEADING ZEROS) * (U SI)
0734A 403E     (06441)        MULRR R3,R7                        ;R3=2*NZ*UI
0734B 3C31     (06442)        LRS R3,1                           ;R3=NZ*UI
0734C EF307286 (06443)        SAF R3,ADC$+W$*ADC$3           ;MOVE TO APS MAIN MEMORY
0734E EF30728A (06444)        SAF R3,ADC$+W$*ADC$4           ;MOVE TO APS MAIN MEMORY
               (06445) ;
               (06446) ; V BUFFER BINDING
               (06447) ; VBASE BINDING
07350 F0620003 (06448)        MOVMR R6,3*H$(R1)                 ;GET V ID
07352 506CFF00 (06449)        MOVKR R6,R6,MSK$LBYT              ;GET PROPER BYTE
07354 3C67     (06450)        LRS R6,7                           ;CONVERT TO BCT$ INDEX
07355 0000     (06451)        EVEN
07356 C04C0582 (06452)        MOVMRL R4,BCT$BA(R6)              ;GET 17-BIT VBASE ADDR
07358 F0707266 (06453)        MOVMR R7,ADC$+W$*ADC$1            ;APS ADDR IN MAIN MEMORY
0735A 564EFFF0 (06454)        IORKR R4,R7,$FFF0
0735C 84407266 (06455)        MOVRML R4,ADC$+W$*ADC$1            ;MOVE TO APS IN MAIN MEMORY
               (06456) ;
0735E C03C0582 (06457)        MOVMRL R3,BCT$BA(R6)              ;GET 17-BIT VBASE ADDR
07360 F05072A6 (06458)        MOVMR R5,ADC$+W$*ADC$10           ;APS ADDR IN MAIN MEMORY
07362 563AFFF0 (06459)        IORKR R3,R5,$FFF0
07364 843072A6 (06460)        MOVRML R3,ADC$+W$*ADC$10           ;MOVE TO APS IN MAIN MEMORY
               (06461) ;
07366 C04C0582 (06462)        MOVMRL R4,BCT$BA(R6)              ;GET 17-BIT VBASE ADDR
07368 F07072D2 (06463)        MOVMR R7,ADC$+W$*ADC$13           ;APS ADDR IN MAIN MEMORY
0736A 564EFFF0 (06464)        IORKR R4,R7,$FFF0
0736C 844072D2 (06465)        MOVRML R4,ADC$+W$*ADC$13           ;MOVE TO APS IN MAIN MEMORY
               (06466) ;
```

SBDCORZ - SPECIAL BINDING MODULE FOR DCORZ

```
                     (06467) ; (VZ-1) BINDING
                     (06468) ; (VZ-1)=VSIZE-1=V WS
07366 F03C0605 (06469)      MOVMR R3,WS+BCT$AD(R6)              ;GET WS = # COLS - 1
07370 E0307269 (06470)      MOVRM R3,ADC$+W$*ADC$1A+W$          ;MOVE TO APS MAIN MEMORY
07372 E03072A9 (06471)      MOVRM R3,ADC$+W$*ADC$10A+W$         ;MOVE TO APS MAIN MEMORY
07374 E03072D5 (06472)      MOVRM R3,ADC$+W$*ADC$13A+W$         ;MOVE TO APS MAIN MEMORY
                     (06473) ;
                     (06474) ; VI BINDING
                     (06475) ; VI=V SAMPLE INCREMENT
07376 F04C0604 (06476)      MOVMR R4,BCT$AD(R6)                 ;R4 = V SI
07378 EF40726A (06477)      SAF R4,ADC$+W$*ADC$1B          ;MOVE TO APS MAIN MEMORY
0737A EF407274 (06478)      SAF R4,ADC$+W$*ADC$1B2         ;MOVE TO APS MAIN MEMORY
0737C EF407292 (06479)      SAF R4,ADC$+W$*ADC$6           ;MOVE TO APS MAIN MEMORY
0737E EF4072AA (06480)      SAF R4,ADC$+W$*ADC$10B         ;MOVE TO APS MAIN MEMORY
07380 EF4072C4 (06481)      SAF R4,ADC$+W$*ADC$12          ;MOVE TO APS MAIN MEMORY
07382 EF4072D6 (06482)      SAF R4,ADC$+W$*ADC$13B         ;MOVE TO APS MAIN MEMORY
                     (06483) ; YBUFFER BINDING
                     (06484) ; YBASE BINDING
                     (06485) ;
07384 F0620001 (06486)      MOVMR R6,W$(R1)                     ;GET Y ID
07386 506CFF00 (06487)      MOVKR R6,R6,MSK$LBYT                ;GET PROPER BYTE
07388 3C67     (06488)      LRS R6,7                            ;CONVERT TO BCT$ INDEX
07389 0000     (06489)      EVEN
0738A C04C0582 (06490)      MOVMRL R4,BCT$BA(R6)                ;GET 17-BIT YBASE ADDR
0738C F07072E8 (06491)      MOVMR R7,ADC$+W$*ADC$16             ;APS ADR IN MAIN MEM
0738E 564FFFF0 (06492)      IORKR R4,R7,$FFF0
07390 844072E8 (06493)      MOVRML R4,ADC$+W$*ADC$16            ;MODIFIED APS INSTR IN MAIN MEM
                     (06494) ;
                     (06495) ; (YZ-1) BINDING
                     (06496) ; YZ-1 = YSIZE - 1 = WS
07392 F07C0605 (06497)      MOVMR R7,W$+BCT$AD(R6)              ;GET WS = # OF COLS - 1
07394 E07072EB (06498)      MOVRM R7,ADC$+W$*ADC$16A+W$         ;MOVE TO APS MAIN MEMORY
                     (06499) ;
                     (06500) ; YI BINDING
                     (06501) ; YI = SAMPLE INCREMENT
07396 F03C0604 (06502)      MOVMR R3,BCT$AD(R6)                 ;R3 = Y SI
07398 EF3072EC (06503)      SAF R3,ADC$+W$*ADC$16B         ;MOVE TO APS MAIN MEMORY
0739A EF3072EE (06504)      SAF R3,ADC$+W$*ADC$17          ;MOVE TO APS MAIN MEMORY
                     (06505) ;
                     (06506) ;
0739C 80000F63 (06507)      JMP AP$BNDR0
```

```
                   (06508) * MPIFF$   MODULK TO PROCESS THE MPIFF(ISA,ISB,FLID) FCB
                   (06509) * KFIELD 9/79
                   (06510) *
                   (06511) *        EXECUTES FUNCTION LIST 'FLID' IF (ISA .NE. 0) .AND.
                   (06512) *                                                (ISB .EQ. 0)
                   (06513) *
                   (06514) *        FCB FORMAT (16 BIT FORMAT SHOWN)
                   (06515) *
                   (06516) *      WORD      LEFT BYTE          RIGHT BYTE
                   (06517) *      ----      ---------          ----------
                   (06518) *       0:        PTR TO NEXT FCB AND FUNCTION LIST FLAG (LSB)
                   (06519) *       1:          105                ISA
                   (06520) *       2:            0                ISB
                   (06521) *       3:            0               FLID
                   (06522) *       4:            0                  0
                   (06523) *       5:            0                  0
                   (06524) *
                   (06525) *
                   (06526) *        PROGRAM REGISTER USAGE:
                   (06527) *        R1:      PTR TO WORD 1 OF FCB ENTRY (SET ON ENTRY)
                   (06528) *        R2:      FUNCTION LIST ID
                   (06529) *        R4:      SCALAR A ID
                   (06530) *        R5:      SCALAR B ID
                   (06531) *
                   (06532) *
                   (06533)          EVEN
073 9E 704200FF    (06534) MPIFF$   MOVWK    R4,R1,MSK$RBYT         ;EXTRACT SCALAR A ID
073A0 F0520001     (06535)          MOVHR    R5,H$(R1)              ;GET SCALAR B ID
073A2 F0220002     (06536)          MOVHR    R2,H$(R1)              ;GET FUNCTION LIST ID
                   (06537) *
073A4 E2080502     (06538)          CMPMZ    ISVT$(R4)              ;SEE IF SCALAR A=0
073A6 011073AA     (06539)          JMP      IFF$1,NEZ              ;IF A .NE. 0, TEST B
073A8 0E70         (06540)          RETURN                          ;IF A .EQ. 0, RETURN
073A9 0800         (06541)          EVEN
                   (06542) *
073AA E20A0502     (06543) IFF$1    CMPMZ    ISVT$(R5)              ;SEE IF SCALAR B=0
073AC 0010192C     (06544)          JMP      XFL$01,EQZ             ;IF B .EQ. 0, GO TO
                   (06545) *                                        ; "EXECUTE FUNCTION LIST"
                   (06546) *                                        ; (FLID IS IN R2).
073AE 0E70         (06547)          RETURN                          ;IF B .NE. 0, RETURN
                   (06548) *
                   (06549) *
073AF 0800         (06550)          EVEN
```

```
                  (06551) ' MPGSC -- G-FLAG SET/CLEAR
                  (06552) ;JJW, 18 OCT 79
                  (06553) ;
                  (06554) ;  G-FLAG NUMBER IN 1(R1)
                  (06555) ;  0/1 = CLEAR/SET IN 2(R1)
                  (06556)         EVEN
073B0 F0220001    (06557) MPGSC:  MOVMR    R2,1(R1)        ;GET G-FLAG NUMBER
073B2 2624        (06558)         INCR     R2,4            ;ADD OFFSET FOR SYS$FLGS
073B3 F0320002    (06559)         MOVMR    R3,2(R1)        ;GET CLR/SET BIT
073B5 3A35        (06560)         LLS      R3,5            ;SHIFT TO PROPER POSITION
073B6 56260020    (06561)         IORKR    R2,R3,$20       ;AND OFF AND PUT IN R2
073B8 E021FFCE    (06562)         MOVRM    R2,SYS$FLGS     ;DO THE MOVE THAT SETS/CLRS IT
073BA 0E70        (06563)         RETURN
073BB 0000        (06564)         EVEN
```

```
                (06565) "DEBUGGING "BREAKPOINT" FACILITY
                (06566) ;
                (06567) ; TO HALT CSPU EXECUTION AND SAVE REGISTERS ON THE STACK,
                (06568) ; PATCH:          "CALL  R1,BKPT"  (AS ASSEMBLED BELOW)
                (06569) ; INTO THE DESIRED BREAKPOINT LOCATION (USING MPLOOK).
                (06570) ;
073BC 061073BE  (06571)          CALL    R1,BKPT
073BE 000073BE  (06572) BKPT:    JMP     BKPT
```

```
                      (06573) *TOP OF EXECUTIVE DEFINITION
                      (06574) *
                      (06575) *
          000073C0    (06576)         TUE$CUR = 0L
                      (06577) *
                      (06578) *
     073C0            (06579)         END
```

TOP OF EXECUTIVE DEFINITION

```
ADC$:       07262 (00221) (06245) (06253) (06356) (06405) (06406) (06415) (06417) (06422) (06428)
                  (06429) (06430) (06431) (06432) (06437) (06443) (06444) (06453) (06455) (06458)
                  (06460) (06463) (06465) (06470) (06471) (06472) (06477) (06478) (06479) (06480)
                  (06481) (06482) (06491) (06493) (06498) (06503) (06504)
ADC$1:      00002 (06257) (06453) (06455)
ADC$10:     00022 (06300) (06316) (06332) (06450) (06460)
ADC$10A:    00023 (06301) (06471)
ADC$10B:    00024 (06302) (06480)
ADC$11:     00026 (06305) (06437)
ADC$12:     00031 (06320) (06481)
ADC$13:     00038 (06328) (06463) (06465)
ADC$13A:    00039 (06329) (06472)
ADC$13B:    0003A (06330) (06482)
ADC$14:     0003C (06333)
ADC$15:     00042 (06254) (06341)
ADC$16:     00043 (06342) (06491) (06493)
ADC$16A:    00044 (06343) (06498)
ADC$16B:    00045 (06344) (06503)
ADC$17:     00046 (06346) (06347) (06504)
ADC$1A:     00003 (06258) (06470)
ADC$1B:     00004 (06259) (06477)
ADC$1B2:    00009 (06267) (06478)
ADC$2:      0000C (06264) (06268) (06272) (06415) (06417)
ADC$2A:     0000D (06273) (06422)
ADC$2B:     0000E (06274) (06428)
ADC$2C:     00010 (06276) (06406)
ADC$3:      00012 (06280) (06443)
ADC$4:      00014 (06282) (06444)
ADC$5:      00017 (06287) (06429)
ADC$6:      00018 (06288) (06479)
ADC$7:      0001D (06294) (06430)
ADC$8:      0001F (06296) (06431)
ADC$9:      00021 (06298) (06432)
ADC$A:      00000 (06240) (06354)
ADC$S:      00005 (06245) (06261) (06405)
ADC$Z:      0009C (06247) (06356)
ADPFA:      065FE (00161) (02807) (02830)
ADPKA$A:    0660A (02801) (02827)
ADPKA$I:    0661A (02797) (02829)
ADPKA$Z:    00020 (02800) (02830)
ADPKAI:     065FE (02808)
ADPKAO:     0000A (02809) (02822)
ADPKU:      065D2 (00160) (02761)
ADPKU$SA:   00000 (02758) (02762) (02788)
ADPKU$SZ:   00012 (02759) (02788)
APDT$ORG:   000E8 (00022) (00124) (00129) (00134) (00139) (00144) (00149) (00154) (00159) (00164)
                  (00169) (00174) (00179) (00184) (00189) (00194) (00199) (00204) (00209) (00214)
                  (00219) (02183) (02186) (02187) (04784) (04787) (04788) (05369) (05372) (05373)
                  (06374) (06377) (06378)
AP$ASSS:    00245 (00023)
AP$BNDR:    00EFA (00024) (03595)
AP$BNDR0:   00F63 (00025) (02283) (05075) (05509) (06507)
AP$BNDR1:   00F2D (00026)
```

TOP OF EXECUTIVE DEFINITION

```
AP$BSL:     0024D (00027) (06393)
AP$CSSC:    00248 (00028)
AP$DONE:    00FB1 (00029) (01664)
AP$DONER:   00FBC (00030)
AP$G0:      0000C (00031) (03597)
AP$G1:      0000D (00032)
AP$LLOC:    00011 (00033) (02202) (04803) (05308)
AP$PBFF:    00010 (00034)
AP$SAID:    0000A (00035)
AP$SSCLR:   0000E (00036)
AP$SSS:     00009 (00037)
APCS$:      06BD0 (00191) (04496) (04765) (04812) (04821) (04824) (04833) (04835) (04845) (04847)
                  (04848) (04850) (04854) (04856) (04857) (04859) (04863) (04865) (04866) (04868)
                  (04872) (04874) (04875) (04877) (04881) (04883) (04884) (04886) (04890) (04892)
                  (04893) (04895) (04899) (04901) (04902) (04904) (04908) (04910) (04911) (04913)
                  (04924) (04926) (04930) (04932) (04933) (04935) (04939) (04941) (04942) (04944)
                  (04948) (04950) (04951) (04953) (04957) (04959) (04960) (04962) (04966) (04968)
                  (04969) (04971) (04982) (04984) (04988) (04990) (05001) (05003) (05007) (05009)
                  (05020) (05022) (05026) (05028) (05029) (05031) (05035) (05037) (05038) (05040)
                  (05044) (05046) (05047) (05049) (05053) (05055) (05056) (05058) (05062) (05064)
                  (05065) (05067) (05071) (05073)
APCS$2:     00037 (04625) (04638)
APCS$3:     0003E (04639) (04652)
APCS$4:     00043 (04633) (04647) (04659) (04665)
APCS$C:     00000 (04494) (04762)
APCS$I:     00002 (04506)
APCS$IR:    00009 (04526) (04982) (04984)
APCS$IS:    00010 (04543) (05001) (05003)
APCS$IT0:   0001A (04568) (05020) (05022)
APCS$IT1:   0001C (04572) (05026) (05028) (05029) (05031)
APCS$IT2:   0001E (04576) (05035) (05037) (05038) (05040)
APCS$IT3:   00020 (04580) (05044) (05046) (05047) (05049)
APCS$IT4:   00022 (04584) (05053) (05055) (05056) (05058)
APCS$IT5:   00024 (04588) (05062) (05064) (05065) (05067)
APCS$MOD:   06BE6 (04510) (04532) (04693)
APCS$O:     0004A (04497) (04688)
APCS$OR:    00050 (04712) (04988) (04990)
APCS$OS:    0005C (04750) (05007) (05009)
APCS$OT0:   00053 (04722) (05071) (05073)
APCS$SA:    00003 (04512) (04812)
APCS$U:     00017 (04559) (04833) (04835)
APCS$V10:   00023 (04586) (04872) (04874) (04875) (04877)
APCS$V11:   00021 (04582) (04881) (04883) (04884) (04886)
APCS$V12:   0001F (04578) (04890) (04892) (04893) (04895)
APCS$V13:   0001D (04574) (04899) (04901) (04902) (04904)
APCS$V14:   0001B (04570) (04908) (04910) (04911) (04913)
APCS$V5:    00026 (04595) (04845) (04847) (04848) (04850)
APCS$V6:    0002C (04605) (04854) (04856) (04857) (04859)
APCS$V9:    00025 (04590) (04863) (04865) (04866) (04868)
APCS$W0:    00035 (04631) (04924) (04926)
APCS$W1:    00032 (04627) (04930) (04932) (04933) (04935)
APCS$W2:    0003C (04645) (04939) (04941) (04942) (04944)
APCS$W3:    00039 (04641) (04948) (04950) (04951) (04953)
```

```
PAGE  150:  [BBND]<DCA16>BBN16N.MSU.2, 29-Dec-80 14:30:40, Ed: WOLF
            TOP OF EXECUTIVE DEFINITION

  APCS$W4:    0041 (04657) (04957) (04959) (04960) (04962)
  APCS$W5:    003E (04653) (04966) (04968) (04969) (04971)
  APCS$Y:     004E (04707) (04821) (04824)
  APCS$Z:     00C8 (04493) (04765)
  APCU$:      6A72 (00190) (04032)
  APCU$B1:    0091 (04350) (04363)
  APCU$B12:   0089 (04315) (04346)
  APCU$B3:    0004 (04323) (04337)
  APCU$CUM:   0095 (04333) (04342) (04359) (04373)
  APCU$SA:    0000 (04029) (04033) (04419)
  APCU$SZ:    00AB (04030) (04419)
  APCU$TOP:   0015 (04003) (04399)
  APS$BMO:    0004 (00038)
  APS$LA:     1FFC0 (00039) (01150) (01670)
  APSLOC:     1FFC0 (00040)
  APSMEM:     1FFC0 (00041) (02712) (04697) (05055)
  APSR:       1FFC0 (00042)
  BCT$AD:     0604 (00044) (02229) (05404) (05421) (05426) (05443) (05460) (05465) (05482) (05499)
                   (05504) (06421) (06426) (06469) (06476) (06497) (06502)
  BCT$AT:     0606 (00045) (03500)
  BCT$BA:     0582 (00046) (02223) (02241) (02251) (02256) (02266) (02271) (04820) (04832) (04844)
                   (04853) (04862) (04871) (04880) (04889) (04898) (04907) (04923) (04929) (04938)
                   (04947) (04956) (04965) (04981) (04987) (05000) (05006) (05019) (05025) (05034)
                   (05043) (05052) (05061) (05070) (05398) (05415) (05437) (05454) (05476) (05493)
                   (06414) (06452) (06457) (06462) (06490)
  BIT$G0:     0000 (00047)
  BIT$RC:     0000 (00048)
  BIT$WS:     0003 (00049) (04823)
  BKPT:       73BE (06571) (06572) (06572)
  CF4DB$:     1AD6 (00051)
  CLR$G0G1:   0792 (00052) (02199) (04806) (05385) (06390)
  COS$:       3190 (00053) (03591) (03788)
  CQ$SB:      0024 (02398) (02401) (02404) (02412)
  CSPU$NOS:   E21FC (00054) (00127) (00132) (00137) (00152) (00157) (00162) (00167) (00172) (00177)
                   (00197) (00207) (00212) (00217)
  CTHC1:      9000 (00239) (00240) (00241) (02478)
  CTHC2:      9810 (00240) (00242) (02482)
  CTHC3:      9000 (00241) (02486)
  CTHDG:      99F0 (00249) (00250) (03492) (03498) (03504)
  CTHG:       9970 (00248) (00249) (03486)
  CTHK1:      9830 (00242) (00243) (01596)
  CTHK2:      9880 (00243) (00244)
  CTHK3:      90F0 (00244) (00245)
  CTHK4:      9910 (00245) (00246)
  CTHK5:      9930 (00246) (00247)
  CTHK6:      9950 (00247) (00248)
  CTHU:       99F8 (00250) (00251) (04629) (04643) (04655)
  D211A5:     0028 (06293) (06312)
  D211A6:     002C (06295) (06313)
  D211A7:     0020 (06297) (06314)
  D211A8:     0030 (06286) (06319) (06321) (06338)
  DCEMD:      0007 (06061) (06114) (06120) (06139) (06147) (06166) (06174) (06193) (06201) (06211)
  DCL0:       0026 (06073) (06102)
```

```
PAGE  151:  [BBND]<DCA16>BBN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            TOP OF EXECUTIVE DEFINITION

  DCL1:       003D (06001) (06129)
  DCL2:       0054 (06009) (06156)
  DCL3:       006B (06097) (06183)
  DCOR$:      7150 (00220) (06051)
  DCOR$SA:    0000 (06048) (06053) (06126) (06153) (06180) (06207) (06213) (06214)
  DCOR$SZ:    0005 (06049) (06214)
  DCOR0:      0009 (06067) (06099)
  DCOR1:      0010 (06075)
  DCOR2:      0017 (06083)
  DCOR3:      001E (06091)
  DEALS$:     70FB (00216) (05937) (05942) (06014)
  DEALS$A:    7132 (05940) (06008)
  DEALS$I:    713A (05936) (06012)
  DEALS$O:    000D (05943) (05967)
  DEALS$S:    000C (05937) (05969)
  DEALS$Z:    0056 (05939) (06014)
  DEALU$:     70E8 (00215) (05913)
  DEALU$SA:   0000 (05910) (05914) (05916) (05920)
  DEALU$SZ:   00E4 (05911) (05920)
  DMY$:       0794 (00056) (01530) (02714) (02715) (05669) (05670) (05857) (05858)
  DUMPS:      6660 (00166) (02890) (02919)
  DUMPS$A:    6672 (02893) (02917)
  DUMPS$I:    667A (02889) (02918)
  DUMPS$Z:    001E (02892) (02919)
  DUMPSI:     0000 (02899)
  DUMPSO:     00E4 (02900) (02907)
  DUMPU:      6620 (00165) (02843)
  DUMPU$SA:   0000 (02840) (02844) (02880)
  DUMPU$SZ:   001C (02841) (02880)
  DVLC1:      9A00 (00251) (00252) (00253) (02479)
  DVLC2:      9A10 (00252) (00254) (02483)
  DVLC3:      9A00 (00253) (02487)
  DVLOG:      9BF0 (00261) (00262) (03493) (03499) (03505)
  DVLG:       9B70 (00260) (00261) (03487)
  DVLK1:      9130 (00254) (00255) (01597)
  DVLK2:      9AB0 (00255) (00256)
  DVLK3:      9AF0 (00256) (00257)
  DVLK4:      9B10 (00257) (00258)
  DVLK5:      9B30 (00258) (00259)
  DVLK6:      9B50 (00259) (00260)
  DVLU:       9BF8 (00262) (00263) (04665)
  DVLUF:      9C00 (00263)
  ERROR$:     01AFA (00058)
  EXMID:      00F0A (02760) (02772)
  F1:         000B (03794) (03797)
  F11A$:      3110 (00060)
  F11AL$:     69F0 (00101) (00106) (03774) (03780) (03869)
  F11AL$3:    001C (03786) (03823)
  F11AL$CA:   6A4C (03777) (03850)
  F11AL$CI:   6A60 (03773) (03866)
  F11AL$SZ:   0080 (03776) (03869)
  F2:         0012 (03802) (03807)
  F4:         002D (03834) (03845) (03851)
```

TOP OF EXECUTIVE DEFINITION

```
FDT$:        07E0 (00061) (00220) (00231)
FEOF$1:      6900 (03574) (03584)
FFOF$SSM:    6906 (00102) (03579)
FEOI$SSM:    6902 (00187) (03573)
FEOS$:       3040 (00062)
FEOSL$:      6920 (00180) (00185) (03612)
FEOSL$1:     0012 (03625) (03642)
FEOSL$2:     0019 (03638) (03653) (03694)
FEOSL$3:     003A (03649) (03701) (03742)
FEOSL$4:     0059 (03697) (03744)
FEOSL$SA:    0000 (03609) (03617) (03754) (03756)
FEOSL$SZ:    0064 (03610) (03756) (03757)
FFT$CBSZ:    079A (00063)
FLG$CLR:     0000 (00064) (03579)
FLG$G0:      0004 (00065) (03573) (03579)
FLG$G1:      0005 (00066)
FLG$G2:      0006 (00067)
FLG$G3:      0007 (00068)
FLG$RI:      0011 (00069)
FLG$SET:     0020 (00070) (03573)
FRMSZ:       0008 (04487) (04520) (04554) (04703) (04715)
GAIN$:       6764 (00175) (03221)
GAIN$DV:     0042 (03267) (03283) (03300) (03317) (03326)
GAIN$Q:      0052 (03250) (03273) (03290) (03307) (03347)
GAIN$Q1:     0055 (03352) (03358)
GAIN$Q2:     005B (03360) (03362)
GAIN$Q3:     005F (03357) (03364)
GAIN$S:      0061 (03227) (03231) (03235) (03373)
GAIN$S1:     0065 (03370) (03396)
GAIN$S2:     0077 (03381) (03403)
GAIN$S3:     0079 (03386) (03406)
GAIN$S4:     007B (03391) (03409)
GAIN$SA:     0000 (03210) (03223) (03421)
GAIN$SZ:     0084 (03219) (03421)
GAINS$:      6874 (00176) (03445) (03454) (03562)
GAINS$1:     0000 (03473) (03480)
GAINS$2:     000C (03475) (03476)
GAINS$A:     68EC (03448) (03556)
GAINS$I:     68F4 (03444) (03560)
GAINS$OP:    0032 (03457) (03534)
GAINS$Q:     0028 (03459) (03514) (03522)
GAINS$S:     0003 (03445) (03462)
GAINS$Z:     008F (03447) (03562)
GATHER$:     1D3C (00072)
GTHRS$:      6E42 (00196) (05140) (05145) (05217)
GTHRS$A:     6E7C (05143) (05211)
GTHRS$I:     6E84 (05139) (05215)
GTHRS$O:     0017 (05146) (05193)
GTHRS$S:     0002 (05140) (05151)
GTHRS$Z:     0056 (05142) (05217)
GTHRU$:      6F32 (00195) (05116)
GTHRU$SA:    0000 (05113) (05117) (05119) (05123)
GTHRU$SZ:    0004 (05114) (05123)
```

TOP OF EXECUTIVE DEFINITION

```
H$:         0001 (0074) (01153) (01644) (01669) (02191) (02196) (02206) (02210) (02213) (02216)
                 (02219) (02229) (02231) (02233) (02234) (02235) (02237) (02247) (02262) (02710)
                 (03504) (03500) (04693) (04700) (04731) (04792) (04797) (04000) (04012) (04016)
                 (04020) (04039) (04917) (04975) (04994) (05013) (05377) (05382) (05394) (05411)
                 (05421) (05422) (05423) (05433) (05450) (05460) (05461) (05462) (05472) (05489)
                 (05499) (05500) (05501) (05853) (06382) (06387) (06401) (06405) (06406) (06410)
                 (06421) (06422) (06448) (06469) (06470) (06471) (06472) (06486) (06497) (06498)
                 (06535)
HFCS$:      6700 (00171) (03103) (03165)
HFCS$1:     0002 (03109)
HFCS$2:     0006 (03115)
HFCS$3:     0010 (03130) (03131)
HFCS$4:     0012 (03132)
HFCS$5:     0016 (03147)
HFCS$6:     001A (03152) (03153)
HFCS$7:     001C (03154)
HFCS$A:     6740 (03089) (03160)
HFCS$I:     6746 (03085) (03164)
HFCS$OP:    0015 (03105) (03144)
HFCS$Z:     005A (03088) (03165)
HFCU$:      6680 (00170) (02971)
HFCU$CR:    0039 (03026) (03027) (03028) (03060)
HFCU$DV:    0028 (02980) (03000) (03040)
HFCU$SA:    0000 (02968) (02976) (03068)
HFCU$SZ:    0040 (02969) (03068)
IBUS:       630B (02010)
IFF$1:      73AA (06539) (06543)
IN:         0000 (00366)
INST:       6202 (01668) (01669) (01675)
INVPS$A:    65BA (02651) (02725)
INVPS$I:    65C2 (02647) (02726)
INVPS$N:    6584 (02667) (02680) (02710)
INVPS$S:    0003 (02648) (02664)
INVPS$Z:    006E (02650) (02727)
INVPSI:     6562 (00156) (02640) (02660) (02727)
INVPSO:     0024 (02661) (02700)
INVPU$:     651A (00155) (02592)
INVPU$SA:   0000 (02589) (02593) (02628)
INVPU$SZ:   0020 (02590) (02628)
ISVT$:      0502 (00076) (02215) (02811) (02823) (06403) (06538) (06543)
ITI:        630A (02017) (02152)
KQ$DONE:    005F (01417) (01423)
KQ$I:       004E (01451) (01589)
KQ$O:       0070 (01589) (01635)
KQ$SUB:     0066 (01591) (01619)
KQUAN:      0053 (01382) (01400)
LOONE:      004F (00434) (00448)
LOAD$AP:    01B7E (00078)
LOAD$AP1:   01B0F (00079)
LOOP:       001D (00395) (00445)
M$$:        0000 (00084) (00507) (00500) (00524) (00525) (00541) (00542) (00567) (00568) (00573)
                 (00574) (00582) (00583) (00700) (00701) (00712) (00713) (00716) (00717) (00743)
                 (00744) (00749) (00750) (01046) (01047) (01048) (01049) (01067) (01068) (01112)
```

```
                  (01113) (01117) (01167) (01168) (01454) (01458) (01459) (01463) (01464) (01510)
                  (01511) (01539) (01540) (01559) (01560) (01594) (01595) (01637) (01630) (01640)
                  (01641) (02467) (02468) (02472) (02515) (02516) (02510) (02519) (02664) (02671)
                  (02672) (02676) (02677) (02681) (02685) (02686) (02689) (02690) (02717) (02718)
                  (02813) (02814) (02909) (02910) (03110) (03111) (03116) (03117) (03148) (03149)
                  (03462) (03463) (03464) (03469) (03470) (03541) (03542) (03546) (03547) (03791)
                  (03792) (03825) (03826) (04513) (04527) (04533) (04544) (04559) (04569) (04571)
                  (04573) (04575) (04577) (04579) (04581) (04583) (04585) (04587) (04589) (04591)
                  (04596) (04606) (04628) (04632) (04642) (04646) (04654) (04658) (04707) (04713)
                  (04723) (04751) (05152) (05153) (05158) (05159) (05171) (05172) (05180) (05181)
                  (05199) (05200) (05284) (05285) (05286) (05289) (05293) (05294) (05295) (05290)
                  (05302) (05303) (05304) (05307) (05321) (05322) (05323) (05326) (05330) (05331)
                  (05332) (05335) (05339) (05340) (05341) (05344) (05625) (05629) (05630) (05635)
                  (05636) (05664) (05665) (05812) (05819) (05820) (05824) (05825) (05829) (05833)
                  (05834) (05860) (05861) (05864) (05865) (05952) (05953) (05970) (05971) (05976)
                  (05977) (05989) (05990) (05998) (05999)
NEWL$:     06316  (00146) (02030) (02177) (02210) (02216) (02224) (02226) (02231) (02233) (02234)
                  (02235) (02242) (02244) (02252) (02254) (02257) (02259) (02267) (02269) (02272)
                  (02274)
NEWL$C:    00000  (02028) (02175)
NEWL$S:    06206  (00145) (01737)
NEWL$SA:   00000  (01734) (01742) (01984) (01986)
NEWL$SZ:   0007E  (01735) (01986)
NEWL$Z:    000A6  (02027) (02177)
NEWL3:     00019  (01774) (01787)
NEWL4:     0001E  (01750) (01814)
NEWL6:     0005D  (01918) (01931)
NEWL7:     00068  (01805) (01946)
NEWLA1:    00002  (02043) (02231)
NEWLA10:   0001A  (02071) (02076)
NEWLA11:   0001D  (02060) (02087) (02210)
NEWLA12:   0001E  (02088)
NEWLA13:   00020  (02091) (02234)
NEWLA15:   00039  (02120) (02128)
NEWLA16:   00043  (02061) (02151) (02216)
NEWLA17:   00046  (02155) (02235)
NEWLA19:   00050  (02165) (02170)
NEWLA2:    00005  (02047) (02252) (02254)
NEWLA3:    00006  (02048) (02267) (02269)
NEWLA4:    00007  (02049) (02233)
NEWLA5:    0000A  (02056) (02117) (02140) (02272) (02274)
NEWLA6:    0000C  (02058) (02257) (02259)
NEWLA7:    0000D  (02059) (02224) (02226)
NEWLA8:    00012  (02065) (02242) (02244)
NEWLI1:    0638C  (02080) (02170)
NPGSC:     073B0  (00232) (06557)
NPIFFS:    0739E  (00229) (06534)
NSK$BMB:   00006  (00081)
NSK$LBVT:  0FF00  (00082) (02192) (02220) (02238) (02248) (02263) (03585) (04793) (04817) (04829)
                  (04910) (04995) (05370) (05395) (05434) (05473) (06383) (06411) (06449) (06487)
NSK$RBVT:  000FF  (00083) (02197) (02207) (02214) (04798) (04809) (04840) (04976) (05014) (05383)
                  (05412) (05451) (05490) (06380) (06402) (06534)
NWLAE:     0001F  (01501) (01507)
```

```
PAGE  155:  [DBND]<DCA16>DDN16M.MSO.2, 29-Dec-80 14:30:40, Ed: WOLF
            TOP OF EXECUTIVE DEFINITION

  NWLF$A:      0610D (01441) (01650)
  NWLF$APS:    060E8 (00141) (01430) (01440) (01650)
  NWLF$APW:    06016 (00140) (01230)
  NWLF$I:      061E2 (01437) (01656)
  NWLF$S:      00002 (01430) (01450) (01454)
  NWLF$SA:     00000 (01234) (01241) (01430)
  NWLF$SZ:     00065 (01235) (01430)
  NWLF$Z:      0010A (01440) (01650)
  NWLFAE:      00039 (01342) (01300)
  NWLFAW:      00051 (01326) (01307)
  NWLFD:       0004D (01367) (01301)
  NWLFRE:      00017 (01277) (01306)
  NWLFRW:      0004F (01256) (01305) (01305)
  NWLFSE:      00010 (01250) (01267)
  NWLOC:       0004R (01561) (01567)
  NWLOD:       0004C (01549) (01569)
  NWLOE:       00032 (01532) (01537)
  NWLOT:       0002A (01499) (01527)
  NWLQ$SSN:    061F2 (00142) (01661)
  NWLSC:       00015 (01475) (01403) (01405)
  NWLSE:       00012 (01476) (01400)
  NWLT:        00020 (01509)
  NWSS$1:      061F8 (01662) (01660)
  NYZ:         06310 (02039) (02042)
  NZ:          00002 (06229) (06261) (06276) (06280) (06282)
  DQE:         0001B (00006) (00656) (00671) (00934) (00943) (02601) (02605) (04056) (04062) (05750)
                     (05754)
  PITS$:       05FA4 (00136) (01023) (01040) (01179)
  PITS$1:      05FE2 (01101) (01116) (01153)
  PITS$A:      06002 (01026) (01174)
  PITS$I:      06000 (01022) (01170)
  PITS$O:      00026 (01043) (01143)
  PITS$S:      00001 (01023) (01046)
  PITS$Z:      00070 (01025) (01179)
  PITSS$A:     070C8 (05800) (05872)
  PITSS$I:     070D6 (05796) (05873)
  PITSS$M:     07498 (05815) (05820) (05853)
  PITSS$S:     00003 (05797) (05812)
  PITSS$Z:     00070 (05799) (05874)
  PITSSI:      07076 (00211) (05797) (05800) (05874)
  PITSSO:      00020 (05809) (05851)
  PITSUS:      0702E (00210) (05741)
  PITSU$SA:    00000 (05738) (05742) (05777)
  PITSU$SZ:    00020 (05739) (05777)
  PITU$:       05FE4 (00135) (00840)
  PITU$0:      00007 (00852) (00860)
  PITU$1:      00010 (00850) (00863)
  PITU$2:      00015 (00858) (00870) (00882)
  PITU$3:      0001B (00854) (00865) (00875)
  PITU$4:      0001D (00856) (00880)
  PITU$5:      0001F (00867) (00886)
  PITU$6:      00025 (00888) (00892)
  PITU$7:      00027 (00891) (00900)
```

TOP OF EXECUTIVE DEFINITION

```
PITU$B:     0040 (00960)
PITU$SA:    0000 (00037) (00844) (00990)
PITU$SZ:    005C (00030) (00998)
PITU$W1:    0037 (00934) (00934)
PITU$W2:    003A (00943) (00943)
L  11$:     035EA (00000)
SB:         0630C (02019) (02007)
SBDCURZ:    072FE (00222) (06374)
SBN$APC:    06C90 (00192) (04703)
SBN$MEWL:   0630D (00147) (02103)
SBN$TN:     06FF4 (00202) (05360)
SBUS:       06300 (02020)
SCL$:       06FA2 (00205) (05553)
SCL$N:      0000C (05560) (05563) (05566) (05573)
SCL$N1:     0000D (05574) (05580)
SCL$SA:     0000 (05550) (05554) (05580)
SCL$SZ:     0019 (05551) (05580)
SCLS$:      06FDC (00206) (05611) (05620) (05604)
SCLS$1:     0000A (05639) (05646)
SCLS$2:     000C (05641) (05642)
SCLS$3:     0019 (05660) (05674)
SCLS$4:     001C (05671) (05672)
SCLS$A:     07014 (05614) (05670)
SCLS$I:     07020 (05610) (05682)
SCLS$OP:    0014 (05622) (05660)
SCLS$S:     0002 (05611) (05625)
SCLS$Z:     005F (05613) (05604)
SHFTL$R5:   007AE (00009)
SINS:       03192 (00091) (03789)
SINCOS$T:   023FA (00090) (03592) (03593)
STAB$:      06444 (00150) (02344)
STAB$OP:    0021 (02461) (02513)
STAB$OT:    F002F (02420) (02427)
STAB$QC:    0017 (02464) (02496) (02504)
STAB$QT:    001B (02304) (02394)
STAB$SA:    0000 (02340) (02345) (02432)
STAB$SZ:    0033 (02341) (02432)
STAB$UN:    0017 (02370) (02372) (02378) (02380) (02387)
STABS$:     06402 (00151) (02440) (02457) (02530)
STABS$A:    06504 (02451) (02520)
STABS$I:    0650C (02447) (02534)
STABS$S:    0007 (02440) (02472)
STABS$Z:    0066 (02450) (02538)
SVT$:       00302 (00092) (02209) (04811)
SVT$UN1:    003CE (00093) (01003) (01592) (02475) (02663) (03485) (03491) (03497) (03503) (04511)
                  (04630) (04644) (04656) (05811)
SYS$FLGS:   1FFCE (00094) (01671) (06562)
TADPK:      F0057 (02792) (02811) (02823)
TENS0:      007B4 (00096)
TNS$:       06FA0 (00201) (05273) (05353) (05399) (05401) (05405) (05406) (05416) (05418) (05422)
                  (05423) (05427) (05428) (05438) (05440) (05444) (05445) (05455) (05457) (05461)
                  (05462) (05466) (05467) (05477) (05479) (05483) (05484) (05494) (05496) (05500)
                  (05501) (05505) (05506)
```

```
THS$C:      000R0 (05271) (05350)
THS$I:      00002 (05281)
THS$U:      00013 (05274) (05310)
THS$HHA:    0001E (05339) (05477) (05479)
THS$HSI1:   00020 (05341) (05483)
THS$HSI2:   00021 (05343) (05484)
THS$SHA:    0000C (05302) (05494) (05496)
THS$SHS1:   0000D (05303) (05500)
THS$SHS2:   0001F (05340) (05501)
THS$SSI1:   0000E (05304) (05505)
THS$SSI2:   0000F (05306) (05506)
THS$UHA:    00002 (05284) (05416) (05418)
THS$UHS1:   00003 (05285) (05422)
THS$UHS2:   00015 (05322) (05423)
THS$USI1:   00004 (05286) (05427)
THS$USI2:   00005 (05288) (05428)
THS$VHA:    00019 (05330) (05438) (05440)
THS$VSI1:   0001B (05332) (05444)
THS$VSI2:   0001C (05334) (05445)
THS$WHA:    00007 (05293) (05455) (05457)
THS$WHS1:   00008 (05294) (05461)
THS$WHS2:   0001A (05331) (05462)
THS$WSI1:   00009 (05295) (05466)
THS$WSI2:   0000A (05297) (05467)
THS$YHA:    00014 (05321) (05399) (05401)
THS$YSI1:   00016 (05323) (05405)
THS$YSI2:   00017 (05325) (05406)
THS$Z:      0004A (05270) (05353)
THU$:       06E9A (04208) (05243)
THU$SA:     00000 (05240) (05244) (05245) (05249)
THU$SZ:     00004 (05241) (05249)
TUE$:       021FE (00C97)
TUE$CUR:    073C0 (00113) (06576)
TUE$PTR:    00200 (00098) (00111)
UB:         06304 (02001) (02065)
UBASE:      0003P (06222) (06272)
UBUS:       06315 (02002)
UI:         00002 (06238) (06274) (06280) (06282) (06287) (06294) (06296) (06298) (06305) (06333)
UKI:        00002 (03782) (03795) (03796) (03810) (03811) (03812) (03813)
UZ:         000FC (06226) (06273)
V3200$:     05DC6 (00126) (00489) (00606)
V3200$G:    0001D (01495) (00562)
V3200$A:    05E22 (01485) (00600)
V3200$I:    05E2A (00481) (00605)
V3200$Z:    00000 (00484) (00606)
VALBUF$:    01C5A (01130)
VB:         06346 (02896) (02047) (02058)
VBASE:      04332 (05223) (06257) (06300) (06328)
VBUS:       06347 (02017)
VI:         00042 (05231) (06259) (06267) (06288) (06302) (06320) (06330)
VKG:        06314 (03710) (00721)
VKY:        03028 (01751) (01757)
VKAS$A:     05E3C (00689) (01765)
```

TOP OF EXECUTIVE DEFINITION

```
 VKASSI:    05FD8 (00685) (00766)
 VKASSZ:    0006C (00688) (00767)
 VKASI:     05F76 (00131) (00698) (00767)
 VKASO:     0041D (00704) (00741)
 VFAUS:     05F40 (00130) (00653)
 VKAUSSA:   0J000 (00650) (00654) (00677)
 VKAUSSZ:   00013 (00651) (00677)
 VLTSVS:    05D82 (00125) (00363)
 VLTSVSSA:  00000 (00360) (00365) (00464)
 VLTSVSSZ:  0005E (00361) (00464) (00465)
 VSNA2S:    02870 (00101)
 VZ:        00005 (06227) (06250) (06301) (06329)
 WS:        00002 (00103) (00124) (00129) (00134) (00139) (00144) (00149) (00154) (00159) (00164)
                  (00169) (00174) (00179) (00184) (00189) (00194) (00199) (00204) (00209) (00214)
                  (00219) (00220) (00231) (00240) (00242) (00243) (00244) (00245) (00246) (00247)
                  (00248) (00249) (00250) (00251) (00252) (00254) (00255) (00256) (00257) (00258)
                  (00259) (00260) (00261) (00262) (00263) (00720) (00722) (00728) (00747) (00752)
                  (00755) (00757) (00759) (01077) (01126) (01127) (01596) (01597) (01620) (01621)
                  (01622) (01624) (01625) (01643) (02184) (02185) (02186) (02187) (02190) (02210)
                  (02216) (02224) (02226) (02231) (02233) (02234) (02235) (02242) (02244) (02252)
                  (02254) (02257) (02259) (02267) (02269) (02272) (02274) (02478) (02479) (02482)
                  (02483) (02486) (02487) (02497) (02498) (02500) (02501) (02678) (02687) (02692)
                  (02693) (02694) (02695) (02696) (02697) (03477) (03486) (03487) (03492) (03493)
                  (03498) (03499) (03504) (03505) (03515) (03516) (03518) (03519) (03593) (04528)
                  (04539) (04545) (04546) (04547) (04548) (04549) (04560) (04566) (04597) (04598)
                  (04599) (04600) (04601) (04607) (04608) (04612) (04629) (04629) (04643) (04643)
                  (04655) (04655) (04666) (04667) (04668) (04714) (04715) (04724) (04725) (04726)
                  (04727) (04735) (04739) (04752) (04753) (04754) (04755) (04756) (04785) (04786)
                  (04787) (04788) (04791) (04812) (04821) (04824) (04833) (04835) (04845) (04847)
                  (04848) (04849) (04850) (04854) (04856) (04857) (04858) (04859) (04863) (04865)
                  (04866) (04867) (04868) (04872) (04874) (04875) (04876) (04877) (04881) (04883)
                  (04884) (04885) (04886) (04890) (04892) (04893) (04894) (04895) (04899) (04901)
                  (04902) (04903) (04904) (04908) (04910) (04911) (04912) (04913) (04924) (04926)
                  (04930) (04932) (04933) (04934) (04935) (04939) (04941) (04942) (04943) (04944)
                  (04948) (04950) (04951) (04952) (04953) (04957) (04959) (04960) (04961) (04962)
                  (04966) (04968) (04969) (04970) (04971) (04982) (04984) (04988) (04990) (05001)
                  (05003) (05007) (05009) (05020) (05022) (05026) (05028) (05029) (05030) (05031)
                  (05035) (05037) (05038) (05039) (05040) (05044) (05046) (05047) (05048) (05049)
                  (05053) (05055) (05056) (05057) (05058) (05062) (05064) (05065) (05066) (05067)
                  (05071) (05073) (05370) (05371) (05372) (05373) (05376) (05399) (05401) (05405)
                  (05406) (05416) (05418) (05422) (05423) (05427) (05428) (05438) (05440) (05444)
                  (05445) (05455) (05457) (05461) (05462) (05466) (05467) (05477) (05479) (05483)
                  (05484) (05494) (05496) (05500) (05501) (05505) (05506) (05643) (05673) (05826)
                  (05836) (05839) (05840) (05841) (05862) (05867) (05955) (06375) (06376) (06377)
                  (06378) (06381) (06405) (06406) (06415) (06417) (06422) (06428) (06429) (06430)
                  (06431) (06432) (06437) (06443) (06444) (06453) (06455) (06458) (06460) (06463)
                  (06465) (06470) (06471) (06472) (06477) (06478) (06479) (06480) (06481) (06482)
                  (06491) (06493) (06498) (06503) (06504) (06536)
 WB:        06300 (02011) (02040) (02056)
 WBUS:      06309 (02012)
 XFLS01:    0192C (00105) (06544)
 YB:        06302 (01996) (02059)
 YBASE:     00064 (06224) (06342)
```

```
                TOP OF EXECUTIVE DEFINITION

  YOUS:         06303 (01999)
  Y1:           00002 (06232) (06344) (06346)
  YR1:          00002 (03703) (03036) (03037) (03047).(03040) (03049) (03050)
  Y2:           0000F (02043) (02049) (02091) (02155) (06220) (06343)
  ZERO:         007DA (00107) (02201) (04002) (05307) (06266) (06312) (06313) (06314) (06319) (06324)
                      (06325) (06326) (06392)

LINES WITH ERRORS:    0  (MAP VERSION 000101.10)   E-    0
```

(#BBN1) ;[BBND]<DCA16>BBN16P.MSD.1, 10-Nov-80 19:15:52, Ed: KFIELD

```
          'CSPI PATCHES TO MAP-300 SNAPII (REL 3.5) EXEC
          *
          * THIS PATCH FILE ASSUMES MAP HAS BEEN LOADED WITH
          * SNAPII EXEC, RELEASE 3.5.
          *
          * THIS PATCH ASSUMES REV. 10 (OR LATER) MICROCODE.
          *
          * DEFINE 'SAF' (SET ADDRESS FIELD) AND 'LAF' (LOAD
          * ADDRESS FIELD) INSTRUCTIONS TO ASSEMBLER.
          * THESE INSTRUCTIONS ARE NEW CSPU INSTRUCTIONS
          * AVAILABLE WITH 'REV 10' CSPU MICROCODE.
          * THEY SET/LOAD THE LOW ORDER 17 BITS OF THE
          * REFERENCED FULL-WORD FROM/INTO THE INDICATED
          * REGISTER.
          *
                  OPADD   SAF,(1 .LS. 14) + (29 .LS. 8) + $EF
                  OPADD   LAF,(1 .LS. 14) + (29 .LS. 8) + $EE
          *
          * DEFINE $0 TO BE CURRENT TOP OF PATCH AREA
          * (UPDATE AS PATCH SPACE IS USED).
          * (SPACE FROM $1C78-$1C97 IS USED BY REL. 3.5)
          *
00001C98          $0 = $1C98
          *
          * DEFINE REL. 3.5 SYMBOLS
          *
00000F63          AP$0NDR0 = $0F63
00000582          BCT$BA = $0582
00000604          BCT$AD = $0604
0000024D          AP$DSL = $024D
0000024A          AP$CSL = $024A
00000E38          SCD$04H= $0E38
00000001          H$ = 1
          *
```

```
                   (00036) "PROBLEM REPORT 01201; NEWSLETTER 023, PAGE 55
                   (00037) *
                   (00038) * PATCH PREBINDING ROUTINES TO WORK WITH PREBINDING
                   (00039) * BUFFERS ABOVE $7FFF (16K).
                   (00040) * (NOTE:  THIS PATCH USES PATCH SPACE.)
                   (00041) *
                   (00042) *
          00000E60 (00043)          0L = $0E60
     00E60 EE3A0582 (00044)          LAF     R3,BCT$BA(R5)
                   (00045) *
          00000F6C (00046)          0L = $0F6C
     00F6C EE440582 (00047)          LAF     R4,UCT$BA(R2)
                   (00048) *
          00000F8C (00049)          0L = $0F8C
     00F8C EF540582 (00050)          SAF     R5,BCT$BA(R2)
                   (00051) *
          00000FB1 (00052)          0L = $0FB1
     00FB1 80001C98 (00053)          JMP     P1201B
                   (00054) P1201A:
                   (00055) *
          00000E58 (00056)          0L = $0E58
     00E58 80001CA0 (00057)          JMP     P1201C
                   (00058) *
          00001C98 (00059)          0L = 00
                   (00060) P1201B:
     01C98 C060024A (00061)          MOVMRL  R6,AP$CSL
     01C9A 9A60FFFF (00062)          ANDIR   R6,$FFFF
     01C9C 9A70FFFF (00063)          ANDIR   R7,$FFFF
     01C9E 80000F83 (00064)          JMP     P1201A
                   (00065) *
                   (00066) P1201C:
     01CA0 9A30FFFF (00067)          ANDIR   R3,$FFFF
     01CA2 80000E38 (00068)          JMP     SCD$04H
                   (00069) *
                   (00070) *
          00001CA4 (00071)          00 = 0L
                   (00072) *
```

```
PAGE    4:  [RBND]<DCA16>RBN15P.MSD.1, 18-Nov-80 19:15:52, Ed: KFIELD
            PROBLEM REPORT #1203; NEWSLETTER #23, PAGE 59

                          (00073) "PROBLEM REPORT #1203; NEWSLETTER #23, PAGE 59
                          (00074) *
                          (00075) * PATCH STANDARD BINDING ROUTINES TO CORRECTLY BIND
                          (00076) * ARRAY FUNCTIONS RESIDING ABOVE 16K ON BUS 1.
                          (00077) * (NOTE:  THIS PATCH USES PATCH SPACE.)
                          (00078) *
            00000F1E      (00079)         $L = $0F1E
      0F1E  00001CA4      (00080)         JMP     P1203B
                          (00081) P1203A:
                          (00082) *
            00001CA4      (00083)         $L = $0
                          (00084) P1203B:
      01CA4 9A60FFFF      (00085)         ANDIR   R6,$FFFF
      01CA6 2661          (00086)         INCR    R6,HS
      01CA7 2671          (00087)         INCR    R7,HS
      01CA8 00000F20      (00088)         JMP     P1203A
                          (00089) *
                          (00090) *
            00001CAA      (00091)         $0 = $L
                          (00092) *


PAGE    5:  [RBND]<DCA16>RBN16P.MSD.1, 18-Nov-80 19:15:52, Ed: KFIELD
            PROBLEM REPORT #3201; NEWSLETTER #23, PAGE 72

                          (00093) "PROBLEM REPORT #3201; NEWSLETTER #23, PAGE 72
                          (00094) *
                          (00095) * PATCH FFT SPECIAL BINDING ROUTINES TO ALLOW FOR
                          (00096) * CORRECT PRE-BINDING OF FFTN.
                          (00097) *
            00005534      (00098)         $L = $5534
      05534 00000F63      (00099)         JMP     AP$BNDR0
                          (00100) *
                          (00101) *
```

```
PAGE    6: [BBND]<DCA16>BBN16P.MSO.1, 10-Nov-80 19:15:52, Ed: KFIELD
           PROBLEM REPORT #3202; NEWSLETTER #24, PAGE 71

                      (00102) 'PROBLEM REPORT #3202; NEWSLETTER #24, PAGE 71
                      (00103) *
                      (00104) * PATCH TO MAKE FFTN PREBINDABLE WITH PREBINDING BUFFER
                      (00105) *  ABOVE $7FFF.
                      (00106) *  (NOTE: THIS PATCH USES PATCH SPACE.)
                      (00107) *
                      (00108) *
                      (00109) *
                      (00110) *
             000054A6 (00111)         $L = $54A6
       054A6 00001CAA (00112)         JMP     P3202B
                      (00113) P3202A:
                      (00114) *
             00001CAA (00115)         $L = $0
                      (00116) P3202B:
       01CAA F0360240 (00117)         MOVHR    R3,AP$BSL
       01CAC 9A30FFFF (00118)         ANDIR    R3,$FFFF
       01CAE 000054A8 (00119)         JMP      P3202A
                      (00120) *
             00001CB0 (00121)         $0 = $L
                      (00122) *


PAGE    7: [BBND]<DCA16>BBN16P.MSO.1, 10-Nov-80 19:15:52, Ed: KFIELD
           PROBLEM REPORT #BBN1; NEWSLETTER #--, PAGE --

                      (00123) 'PROBLEM REPORT #BBN1; NEWSLETTER #--, PAGE --
                      (00124) *
                      (00125) * PATCH TO EXEC UTILITY 'VALBUFS' TO ALLOW FOR
                      (00126) *  CORRECT VALIDATION OF BUFFERS WITH NEGATIVE
                      (00127) *  SAMPLE INCREMENTS (SI'S).
                      (00128) *  KFIELD 7/80
                      (00129) *
             00001C5E (00130)         $L = $1C5E
       01C5E F0340604 (00131)         MOVHR    R3,BCT$AD(R2)          ;LOAD VECTOR SI
                      (00132) *
             00001C62 (00133)         $L = $1C62
       01C62 C2340605 (00134)         MULHR    R3,BCT$AD+H$(R2)       ;MULT BY SIZE-1
                      (00135) *
       01C64          (00136)         END
```

```
PAGE     0:  [BBNO]<DCA16>BBN16P.MSO.1, 10-Nov-80 19:15:52, Ed: KFIELD
             PROBLEM REPORT (BBN1) NEWSLETTER #--, PAGE --

  AP$ANDR0:   00F63 (00028) (00099)
  AP$BSL:     0074D (00031) (00117)
  AP$CSL:     0024A (00032) (00061)
  BCT$AD:     00604 (00030) (00131) (00134)
  BCT$BA:     00582 (00029) (00044) (00047) (00050)
  N$:         00F01 (00034) (00086) (00087) (00134)
  P1201A:     00F83 (00054) (00064)
  P1201B:     01C98 (00053) (00060)
  P1201C:     01CA0 (00057) (00066)
  P1203A:     00F20 (00001) (00088)
  P1203B:     01CA4 (00080) (00084)
  P3202A:     054A0 (00113) (00119)
  P3202B:     01CAA (00112) (00116)
  SCD$04N:    00E3B (00033) (00068)

LINES WITH ERRORS:     0  (MAP VERSION 000101.10)   E-     0
```

PAGE 1: [BBND]<DCA16>BBN16T.MSG.1, 10-Nov-80 19:15:02, Ed: KFIELD

(00001) ;[BBND]<DCA16>BBN16T.MSG.1, 10-Nov-80 19:15:02, Ed: KFIELD

```
                    (00002) ^CODING AND DECODING TABLES
                    (00003) ;
                    (00004) ;
                    (00005) ;CODING TABLES GIVE THE BOUNDARIES OR THRESHOLDS BETWEEN
                    (00006) ;  INTERVALS CORRESPONDING TO THE CODES.  THE RULE FOR QUANTIZATION
                    (00007) ;  AND CODING IS TO TEST THE VALUE TO BE QUANTIZED AGAINST CTH(I),
                    (00008) ;  I=0,1,...,LENGTH-1.  THE FIRST I FOR WHICH VALUE<CTH(I) IS
                    (00009) ;  THE CODE (AND WE DECODE IT AS DVL(I)).
                    (00010) ;
          00009800  (00011) .L=$9800               ;THIS IS WHERE THEY GO
                    (00012)
                    (00013) CTHC1:
09800 B6A7EFC0 (00014) CTHC3:  DATA -.42700000    ;CODING TABLE FOR PITCH-TAPS C1 AND C3 (8)
09802 A70A3D40 (00015)         DATA -.30500000
09804 976C8B40 (00016)         DATA -.18300000
09806 FCED913F (00017)         DATA -.06100000
09808 7CED913F (00018)         DATA  .06100000
0980A 176C8B40 (00019)         DATA  .18300000
0980C 270A3D40 (00020)         DATA  .30500000
0980E 36A7EFC0 (00021)         DATA  .42700000
09810 F10A3D40 (00022) CTHC2:  DATA -.88312500    ;C2 CODING TABLE (16)
09812 E87AE140 (00023)         DATA -.81625000
09814 DFEB84C0 (00024)         DATA -.74937499
09816 D75C28C0 (00025)         DATA -.68249999
09818 CECCCCC0 (00026)         DATA -.61562499
0981A C63D7040 (00027)         DATA -.54874998
0981C BDAE1440 (00028)         DATA -.48187498
0981E B51EB840 (00029)         DATA -.41499998
09820 AC8F5BC0 (00030)         DATA -.34812498
09822 A3FFFFC0 (00031)         DATA -.28124999
09824 9B70A3C0 (00032)         DATA -.21437499
09826 92E147C0 (00033)         DATA -.14749999
09828 8A51EB40 (00034)         DATA -.08062499
0982A 9C28F43F (00035)         DATA -.01374999
0982C 6CCCCF3F (00036)         DATA  .05312502
0982E 0F5C28C0 (00037)         DATA  .12000000
                    (00038)
09830 FE219BC0 (00039) CTHK1:  DATA -.98540065    ;CODING TABLE FOR K1 (64)
09832 FDE5FBC0 (00040)         DATA -.98358107
09834 FDA2FF40 (00041)         DATA -.98153679
09836 FD57C1C0 (00042)         DATA -.97924066
09838 FD034540 (00043)         DATA -.97666232
0983A FCA46DC0 (00044)         DATA -.97376801
0983C FC3A0040 (00045)         DATA -.97052009
0983E FBC29E40 (00046)         DATA -.96687680
09840 FB3CC240 (00047)         DATA -.96279173
09842 FAA6BDC0 (00048)         DATA -.95821358
09844 F9FE0540 (00049)         DATA -.95308557
09846 F9429840 (00050)         DATA -.94734517
09848 F8702FC0 (00051)         DATA -.94092366
0984A F784F9C0 (00052)         DATA -.93374563
0984C F67F4840 (00053)         DATA -.92572883
0984E F5592040 (00054)         DATA -.91678173
```

```
09850 F4127640 (00055)         DATA -.90681341
09852 F2A6BDC0 (00056)         DATA -.89571356
09854 F1125BC0 (00057)         DATA -.88337240
09856 EF5162C0 (00058)         DATA -.86967121
09858 ED5FC1C0 (00059)         DATA -.85448478
0985A EB392D40 (00060)         DATA -.83768241
0985C E8D93940 (00061)         DATA -.81912912
0985E E63B6440 (00062)         DATA -.79868747
09860 E35B24C0 (00063)         DATA -.77621973
09862 E0342040 (00064)         DATA -.75159076
09864 DCC207C0 (00065)         DATA -.72467132
09866 D900F840 (00066)         DATA -.69534210
09868 D4ED8240 (00067)         DATA -.66349818
0986A D084D840 (00068)         DATA -.62905408
0986C CBC4FD40 (00069)         DATA -.59194915
0986E C6ACF340 (00070)         DATA -.55215304
09870 C13CE7C0 (00071)         DATA -.50967117
09872 BB765D40 (00072)         DATA -.46454968
09874 B55C5140 (00073)         DATA -.41687981
09876 AEF354C0 (00074)         DATA -.36680087
09878 A8419940 (00075)         DATA -.31450189
0987A A14EEE40 (00076)         DATA -.26022125
0987C 9A24AE40 (00077)         DATA -.20424442
0987E 92CD9940 (00078)         DATA -.14689926
09880 8A5595C0 (00079)         DATA -.08854935
09882 BC9737BF (00080)         DATA -.02958530
09884 3C97413F (00081)         DATA 0.02958537
09886 0A559640 (00082)         DATA 0.08854942
09888 12CD98C0 (00083)         DATA 0.14689934
0988A 1A24AF40 (00084)         DATA 0.20424450
0988C 214EEEC0 (00085)         DATA 0.26022132
0988E 2B4199C0 (00086)         DATA 0.31450195
09890 2EF35540 (00087)         DATA 0.36680095
09892 355C51C0 (00088)         DATA 0.41687988
09894 3B765DC0 (00089)         DATA 0.46454975
09896 413CE840 (00090)         DATA 0.50967123
09898 46ACF3C0 (00091)         DATA 0.55215311
0989A 4BC4FDC0 (00092)         DATA 0.59194921
0989C 5084DAC0 (00093)         DATA 0.62905414
0989E 54ED8240 (00094)         DATA 0.66349822
098A0 5900F8C0 (00095)         DATA 0.69534215
098A2 5CC207C0 (00096)         DATA 0.72467136
098A4 60342040 (00097)         DATA 0.75159079
098A6 635B2A40 (00098)         DATA 0.77621976
098A8 663A6440 (00099)         DATA 0.79868749
098AA 68D93940 (00100)         DATA 0.81912915
098AC 6A392DC0 (00101)         DATA 0.83768244
098AE 6D5FC1C0 (00102)         DATA 0.85448480
098B0 DD91C340 (00103) CTHK2:  DATA -.73101083     ;CODING TABLE FOR K2 (32)
098B2 D8741040 (00104)         DATA -.69104202
098B4 D2BA84C0 (00105)         DATA -.64634765
098B6 CC61A240 (00106)         DATA -.59672958
098B8 C5631540 (00107)         DATA -.54208627
```

CODING AND DECODING TABLES

```
0980A BDC08DC0 (00108)           DATA -.48243076
098BC B57F85C0 (00109)           DATA -.41795417
098BE ACAAD740 (00110)           DATA -.34896366
098C0 A3530640 (00111)           DATA -.27597119
098C2 998F2540 (00112)           DATA -.19965046
098C4 8F7749C0 (00113)           DATA -.12002789
098C6 D2DA503F (00114)           DATA -.04045165
098C8 52D8443F (00115)           DATA 0.04045156
098CA 0F774940 (00116)           DATA 0.12082781
098CC 198E24C0 (00117)           DATA 0.19965037
098CE 27530540 (00118)           DATA 0.27597111
098D0 2CAAD6C0 (00119)           DATA 0.34896358
098D2 357F84C0 (00120)           DATA 0.41795409
098D4 3DC08CC0 (00121)           DATA 0.48243069
098D6 456314C0 (00122)           DATA 0.54200621
098D8 4C61A1C0 (00123)           DATA 0.59672952
098DA 528B84C0 (00124)           DATA 0.64634760
098DC 58741040 (00125)           DATA 0.69104198
098DE 5D91C2C0 (00126)           DATA 0.73101079
098E0 671D6BC0 (00127)           DATA 0.76652286
098E2 662163C0 (00128)           DATA 0.79789401
098E4 69A8E240 (00129)           DATA 0.82546644
098E6 6CBF6DC0 (00130)           DATA 0.84959197
098E8 6F707140 (00131)           DATA 0.87061901
098EA 71C6EBC0 (00132)           DATA 0.88880300
098EC 73CD37C0 (00133)           DATA 0.90470025
098EE 758CECC0 (00134)           DATA 0.91836321
098F0 DCC263C0 (00135) CTHK3:    DATA -.72460231     ;CODING TABLE FOR K3 (16)
098F2 D4A54FC0 (00136)           DATA -.66129490
098F4 CB1C9640 (00137)           DATA -.58680991
098F6 C01C6240 (00138)           DATA -.50086620
098F8 B3AE8240 (00139)           DATA -.40376312
098FA A5F7B940 (00140)           DATA -.29662241
098FC 9739EE40 (00141)           DATA -.18145542
098FE FD1ACABF (00142)           DATA -.06108626
09900 7D1ACABF (00143)           DATA 0.06108626
09902 1739EE40 (00144)           DATA 0.18145541
09904 25F7B940 (00145)           DATA 0.29662241
09906 33AE8240 (00146)           DATA 0.40376312
09908 401C6240 (00147)           DATA 0.50086619
0990A 4B1C9640 (00148)           DATA 0.58680990
0990C 54A54F40 (00149)           DATA 0.66129489
0990E 5CC263C0 (00150)           DATA 0.72460230
09910 C4292040 (00151) CTHK4:    DATA -.53250506     ;CODING TABLE FOR K4 (16)
09912 B9B3C0C0 (00152)           DATA -.45079815
09914 AE2DA040 (00153)           DATA -.36076742
09916 A1B96740 (00154)           DATA -.26347055
09918 948B48C0 (00155)           DATA -.16050062
0991A EF6ABABF (00156)           DATA -.05391451
0991C 6E6ABABF (00157)           DATA 0.05391451
0991E 148B48C0 (00158)           DATA 0.16050061
09920 21B966C0 (00159)           DATA 0.26347053
09922 2E2DA040 (00160)           DATA 0.36076741
```

```
09924 39B3C0C0 (00161)         DATA 0.45079814
09926 44292040 (00162)         DATA 0.53250505
09928 4D7C8640 (00163)         DATA 0.60536267
0992A 55AC2340 (00164)         DATA 0.66931576
0992C 5CC263C0 (00165)         DATA 0.72460230
0992E 62D25C40 (00166)         DATA 0.77204470
09930 BC9966C0 (00167) CTHK5:  DATA -.47343143      ;CODING TABLE FOR K5 (16)
09932 B2E897C0 (00168)         DATA -.39772316
09934 A878A340 (00169)         DATA -.31618157
09936 9D664CC0 (00170)         DATA -.22968443
09938 91D7DD40 (00171)         DATA -.13940015
0993A DFB7BF3F (00172)         DATA -.04673719
0993C 5FB7BCBF (00173)         DATA 0.04673717
0993E 11D7DD40 (00174)         DATA 0.13940014
09940 1D664CC0 (00175)         DATA 0.22968442
09942 2878A340 (00176)         DATA 0.31618155
09944 32E897C0 (00177)         DATA 0.39772316
09946 3C9966C0 (00178)         DATA 0.47343142
09948 45786240 (00179)         DATA 0.54273631
0994A 4D7C8640 (00180)         DATA 0.60536267
0994C 54A54F40 (00181)         DATA 0.66129489
0994E 5AF92440 (00182)         DATA 0.71072024
09950 A9578440 (00183) CTHK6:  DATA -.32290331      ;CODING TABLE FOR K6 (16)
09952 9E0E0BC0 (00184)         DATA -.23480363
09954 924022C0 (00185)         DATA -.14250225
09956 E1EDDFBF (00186)         DATA -.04781699
09958 61EDDCBF (00187)         DATA 0.04781697
0995A 124022C0 (00188)         DATA 0.14250224
0995C 1E0E0B40 (00189)         DATA 0.23480362
0995E 29578440 (00190)         DATA 0.32290330
09960 33F43840 (00191)         DATA 0.40589048
09962 3DC65740 (00192)         DATA 0.48261537
09964 46081BC0 (00193)         DATA 0.55258511
09966 4ECA3640 (00194)         DATA 0.61554603
09968 55F46A40 (00195)         DATA 0.67152148
0996A 5C41B940 (00196)         DATA 0.72075575
0996C 61BF6940 (00197)         DATA 0.76365393
0996E 667F2940 (00198)         DATA 0.80072517
               (00199) CTHG:                            ;CODING TABLE FOR (OVERALL) ENERGY (64)
09970 41FB90BA (00200)         DATA    0.0000000307255794
09972 5085E13A (00201)         DATA    0.0000000375837437
09974 6289C33A (00202)         DATA    0.0000000459727088
09976 78C3083A (00203)         DATA    0.0000000562341396
09978 093B77BB (00204)         DATA    0.0000000687859929
0997A 0B4B033B (00205)         DATA    0.0000000841395282
0997C 0DD94E39 (00206)         DATA    0.0000001029200590
0997E 10E5A2BB (00207)         DATA    0.0000001258925660
09980 14AB25BB (00208)         DATA    0.0000001539926070
09982 194B2B3B (00209)         DATA    0.0000001883649330
09984 1EECCDBB (00210)         DATA    0.0000002304093330
09986 25D3E43B (00211)         DATA    0.0000002818383390
09988 2E4567BB (00212)         DATA    0.0000003447466740
0998A 3899623B (00213)         DATA    0.0000004216965940
```

```
0998C 453B043B (00214)         DATA    0.0000005158222220
0998E 54AF000B (00215)         DATA    0.0000006309575000
09990 67960E0B (00216)         DATA    0.0000007717916330
09992 7E85B03B (00217)         DATA    0.0000009440610360
09994 09AFE03C (00218)         DATA    0.0000011547822100
09996 00D966BC (00219)         DATA    0.0000014125376900
09998 0E7F7ADC (00220)         DATA    0.0000017278263900
0999A 11BAAF3C (00221)         DATA    0.0000021134093600
0999C 15AFC03C (00222)         DATA    0.0000025852352200
0999E 1A06F0BC (00223)         DATA    0.0000031622782000
099A0 2072B9BC (00224)         DATA    0.0000038681208700
099A2 2700D93C (00225)         DATA    0.0000047315136300
099A4 308CD1BC (00226)         DATA    0.0000057876205800
099A6 3B6305BC (00227)         DATA    0.0000070794589000
099A8 40A471BC (00228)         DATA    0.0000086596445500
099AA 580B4D3C (00229)         DATA    0.0000105925379000
099AC 6CB0AA3C (00230)         DATA    0.0000129568696000
099AE 084F353D (00231)         DATA    0.0000158489336000
099B0 0A2B043D (00232)         DATA    0.0000193865209000
099B2 0C6ECDBD (00233)         DATA    0.0000237137406000
099B4 0F353A3D (00234)         DATA    0.0000290068133000
099B6 129A390D (00235)         DATA    0.0000354813456000
099B8 16C1303D (00236)         DATA    0.0000434010303000
099BA 1BD5693D (00237)         DATA    0.0000530884504000
099BC 220BDA3D (00238)         DATA    0.0000649381709000
099BE 29A54B3D (00239)         DATA    0.0000794328262000
099C0 32F0F8BD (00240)         DATA    0.0000971628124000
099C2 3E4FCEBD (00241)         DATA    0.0001188502310000
099C4 4C3B5D3D (00242)         DATA    0.0001453784520000
099C6 5D3BAA3D (00243)         DATA    0.0001778279600000
099C8 720B1BBD (00244)         DATA    0.0002175204340000
099CA 0007FA3E (00245)         DATA    0.0002660725480000
099CC 0AAA2BBE (00246)         DATA    0.0003254617990000
099CE 0D0B90BE (00247)         DATA    0.0003981072000000
099D0 0FF4FABE (00248)         DATA    0.0004869675720000
099D2 1384C6BE (00249)         DATA    0.0005956622120000
099D4 17E017BE (00250)         DATA    0.0007286182370000
099D6 1D345ABE (00251)         DATA    0.0008912509730000
099D8 23B9213E (00252)         DATA    0.0010901845700000
099DA 2B0263BE (00253)         DATA    0.0013335215500000
099DC 357345BE (00254)         DATA    0.0016311730900000
099DE 416179BE (00255)         DATA    0.0019952625400000
099E0 4FF9653E (00256)         DATA    0.0024406191100000
099F2 61D3343E (00257)         DATA    0.0029853827400000
099E4 77A906BE (00258)         DATA    0.0036517415100000
099E6 0925E0BF (00259)         DATA    0.0044668363400000
099E8 0B30A3BF (00260)         DATA    0.0054638661100000
099EA 0DB00C3F (00261)         DATA    0.0066834391900000
099FC 100F2C0F (00262)         DATA    0.0081752307000000
099EE 147AE13F (00263)         DATA    0.0099999998900000
099F0 37DFC040 (00264) CTHDG:  DATA 0.43651583    ;CODING TABLE FOR DELTA-GAIN (4)
099F2 721482C0 (00265)         DATA 0.89125094
099F4 0D460541 (00266)         DATA 1.65958689
```

```
09F6 1F400043 (00267)         DATA 1000.0000000
099F8 FD34C740 (00268) CTNU:   DATA -0.97817319     ;CODING TABLE FOR RESIDUAL (4)
099FA 00000000 (00269)         DATA 0.00000000
099FC 7D34C740 (00270)         DATA 0.97817319
099FE 1F400043 (00271)         DATA 1000.0000000
               (00272) ;
               (00273) ;DECODING TABLES GIVE THE CORRESPONDENCE BETWEEN CODE (AN
               (00274) ;  INTEGER) AND QUANTIZED VALUE (FLOATING POINT).
               (00275) ;
               (00276) DVLC1:
09A00 BE76CBC0 (00277) DVLC3:  DATA -.48000000     ;DECODING TABLE FOR C1 AND C3 (8)
09A02 AED916C0 (00278)         DATA -.36600000
09A04 9F3B6440 (00279)         DATA -.24400000
09A06 8F9DB240 (00280)         DATA -.12200000
09A08 00000000 (00281)         DATA  .00000000
09A0A 0F9DB240 (00282)         DATA  .12200000
09A0C 1F3B6440 (00283)         DATA  .24400000
09A0E 2ED916C0 (00284)         DATA  .36600000
09A10 F551EBC0 (00285) DVLC2:  DATA -.91656251     ;DECODING TABLE FOR C2 (16)
09A12 ECC28F40 (00286)         DATA -.84968750
09A14 E4333340 (00287)         DATA -.78281250
09A16 DBA3D740 (00288)         DATA -.71593750
09A18 D3147AC0 (00289)         DATA -.64906249
09A1A CA851EC0 (00290)         DATA -.58218749
09A1C C1F5C240 (00291)         DATA -.51531249
09A1E B9666640 (00292)         DATA -.44843748
09A20 B0D70A40 (00293)         DATA -.38156248
09A22 A847ADC0 (00294)         DATA -.31468748
09A24 9FB851C0 (00295)         DATA -.24781249
09A26 9728F5C0 (00296)         DATA -.18093749
09A28 8E9999C0 (00297)         DATA -.11406249
09A2A E0A3D58F (00298)         DATA -.04718749
09A2C 2851EE3F (00299)         DATA  .01968752
09A2E 0A147B40 (00300)         DATA  .08656252
               (00301) ;
09A30 FE3CE9C0 (00302) DVLK1:  DATA -.98623391     ;DECODING TABLE FOR K1 (64)
09A32 FE04AA40 (00303)         DATA -.98451736
09A34 FDC576C0 (00304)         DATA -.98258864
09A36 FD7E7840 (00305)         DATA -.98042203
09A38 FD2EBC40 (00306)         DATA -.97798879
09A3A FCD537C0 (00307)         DATA -.97525692
09A3C FC700F40 (00308)         DATA -.97219076
09A3E FC000540 (00309)         DATA -.96875066
09A40 FB819940 (00310)         DATA -.96489257
09A42 FAF3E1C0 (00311)         DATA -.96056771
09A44 FA5519C0 (00312)         DATA -.95572206
09A46 F9A34D40 (00313)         DATA -.95029607
09A48 F8DC55C0 (00314)         DATA -.94422411
09A4A F7FDD7C0 (00315)         DATA -.93743417
09A4C F7053DC0 (00316)         DATA -.92984749
09A4E F5EF0940 (00317)         DATA -.92137819
09A50 F48A38C0 (00318)         DATA -.91193306
09A52 F36173C0 (00319)         DATA -.90141150
```

```
09A54 F1E1DDC0 (00320)         DATA -.88970540
09A56 F037B040 (00321)         DATA -.87669940
09A58 EE5EEB40 (00322)         DATA -.86227170
09A5A EC535C40 (00323)         DATA -.84629394
09A5C EA10A6C0 (00324)         DATA -.82863319
09A5E E7925340 (00325)         DATA -.80915297
09A60 E4D3DBC0 (00326)         DATA -.78771539
09A62 E100C4C0 (00327)         DATA -.76418365
09A64 DEB4B640 (00328)         DATA -.73842508
09A66 DAEB98C0 (00329)         DATA -.71031488
09A68 D7010840 (00330)         DATA -.67974033
09A6A D2C3F8C0 (00331)         DATA -.64660559
09A6C CE2FE740 (00332)         DATA -.61003690
09A6E C9440340 (00333)         DATA -.57238807
09A70 C3FFDDC0 (00334)         DATA -.53124595
09A72 BE644BC0 (00335)         DATA -.48743577
09A74 B87307C0 (00336)         DATA -.44102574
09A76 B23157C0 (00337)         DATA -.39213083
09A78 ABA31840 (00338)         DATA -.34091512
09A7A A4CFD4C0 (00339)         DATA -.28759249
09A7C 9DC01CC0 (00340)         DATA -.23242530
09A7E 967E05C0 (00341)         DATA -.17572091
09A80 8F14EC40 (00342)         DATA -.11782601
09A82 F913520F (00343)         DATA -.05911089
09A84 000F0000 (00344)         DATA 0.00000000
09A86 79135C3F (00345)         DATA 0.05911896
09A88 0F14ECC0 (00346)         DATA 0.11782609
09A8A 167F0640 (00347)         DATA 0.17572098
09A8C 1DC01D40 (00348)         DATA 0.23242537
09A8E 24CFD540 (00349)         DATA 0.28759256
09A90 2BA318C0 (00350)         DATA 0.34091519
09A92 32315A40 (00351)         DATA 0.39213091
09A94 387308C0 (00352)         DATA 0.44102581
09A96 3E644C40 (00353)         DATA 0.48743584
09A98 43FFDE40 (00354)         DATA 0.53124601
09A9A 49440340 (00355)         DATA 0.57238812
09A9C 4E2FE7C0 (00356)         DATA 0.61003695
09A9E 52C3F940 (00357)         DATA 0.64660564
09AA0 57010840 (00358)         DATA 0.67974030
09AA2 5AEB9940 (00359)         DATA 0.71031493
09AA4 5EB4B6C0 (00360)         DATA 0.73842513
09AA6 61D0C540 (00361)         DATA 0.76418369
09AA8 64D3DBC0 (00362)         DATA 0.78771544
09AAA 67925340 (00363)         DATA 0.80915300
09AAC 6A10A740 (00364)         DATA 0.82863321
09AAE 6C535C40 (00365)         DATA 0.84629396
09AB0 DFE93A40 (00366) DVLK2:  DATA -.74930506     ;DECODING TABLE FOR K2 (32)
09AB2 DB15C340 (00367)         DATA -.71160168
09AB4 D5AB9E40 (00368)         DATA -.66929988
09AB6 CFA30940 (00369)         DATA -.62216299
09AB8 C8F700C0 (00370)         DATA -.57003794
09ABA C1A617C0 (00371)         DATA -.51280126
09ABC B9B34DC0 (00372)         DATA -.45078440
```

PAGE 9: [BBND]<DCA16>BWN16T.MSD.1, 10-Nov-80 19:15:02, Ed: KFIELD
CODING AND DECODING TABLES

```
09ABE B1260D40 (00373)         DATA -.38399474
09AC0 A80F1240 (00374)         DATA -.31292941
09AC2 9E7CA1C0 (00375)         DATA -.23817849
09AC4 948019C0 (00376)         DATA -.16049504
09AC6 8A56B4C0 (00377)         DATA -.08077100
09AC8 00000000 (00378)         DATA 0.00000000
09ACA 0A56B440 (00379)         DATA 0.08077099
09ACC 14801940 (00380)         DATA 0.16049496
09ACE 1E7CA140 (00381)         DATA 0.23817841
09AD0 280F1140 (00382)         DATA 0.31292932
09AD2 31260CC0 (00383)         DATA 0.38399466
09AD4 39834CC0 (00384)         DATA 0.45070433
09AD6 41A61740 (00385)         DATA 0.51288119
09AD8 48F70040 (00386)         DATA 0.57003786
09ADA 4FA300C0 (00387)         DATA 0.62216294
09ADC 55AB9DC0 (00388)         DATA 0.66929983
09ADE 5B15C340 (00389)         DATA 0.71160164
09AE0 5FE939C0 (00390)         DATA 0.74930502
09AE2 642FADC0 (00391)         DATA 0.78270507
09AE4 67F3FA40 (00392)         DATA 0.81213313
09AE6 6B418DC0 (00393)         DATA 0.83793808
09AE8 6E23EDC0 (00394)         DATA 0.86047144
09AEA 70A655C0 (00395)         DATA 0.88007615
09AEC 72D37940 (00396)         DATA 0.89707866
09AEE 748553C0 (00397)         DATA 0.91170370
09AF0 E0509040 (00398) DVLK3:  DATA -.75245864    ;DECODING TABLE FOR K3 (16)
09AF2 D8DFE640 (00399)         DATA -.69433287
09AF4 D00F9340 (00400)         DATA -.62547535
09AF6 C5CB6340 (00401)         DATA -.54526938
09AF8 BA118940 (00402)         DATA -.45366014
09AFA ACF8CAC0 (00403)         DATA -.35134251
09AFC 9EB47A40 (00404)         DATA -.23988272
09AFE 8F947740 (00405)         DATA -.12171834
09B00 00000000 (00406)         DATA 0.00000000
09B02 0F947740 (00407)         DATA 0.12171833
09B04 1EB47A40 (00408)         DATA 0.23988271
09B06 2CF8CA40 (00409)         DATA 0.35134250
09B08 3A1188C0 (00410)         DATA 0.45366013
09B0A 45CB62C0 (00411)         DATA 0.54526937
09B0C 500F9340 (00412)         DATA 0.62547535
09B0E 58DFE640 (00413)         DATA 0.69433286
09B10 C0F70340 (00414) DVLK4:  DATA -.57005347    ;DECODING TABLE FOR K4 (16)
09B12 BF1200C0 (00415)         DATA -.49273689
09B14 B4110A40 (00416)         DATA -.40677000
09B16 A80F6940 (00417)         DATA -.31293977
09B18 9B35ADC0 (00418)         DATA -.21257565
09B1A 8DC319C0 (00419)         DATA -.10751649
09B1C 00000000 (00420)         DATA 0.00000000
09B1E 0DC319C0 (00421)         DATA 0.10751648
09B20 1B35ADC0 (00422)         DATA 0.21257564
09B22 280F6940 (00423)         DATA 0.31293977
09B24 34110A40 (00424)         DATA 0.40677000
09B26 3F1200C0 (00425)         DATA 0.49273689
```

```
09B28 40F78340 (00426)         DATA 0.57005347
09B2A 51B065C0 (00427)         DATA 0.63843988
09B2C 59596740 (00428)         DATA 0.69884088
09B2E 5FE9B940 (00429)         DATA 0.74932021
09B30 C123E640 (00430) DVLK5:  DATA -.50890009     ;DECODING TABLE FOR K5 (16)
09B32 B7DA4B40 (00431)         DATA -.43634928
09B34 ADC6F240 (00432)         DATA -.35763384
09B36 A3019540 (00433)         DATA -.27340579
09B38 97A0DDC0 (00434)         DATA -.18493246
09B3A 8BF84AC0 (00435)         DATA -.09327063
09B3C 00000000 (00436)         DATA 0.00000000
09B3E 0BF04AC0 (00437)         DATA 0.09327062
09B40 17A0DDC0 (00438)         DATA 0.18493246
09B42 23019540 (00439)         DATA 0.27340579
09B44 2DC6F1C0 (00440)         DATA 0.35763383
09B46 370A4B40 (00441)         DATA 0.43634928
09B48 4123E640 (00442)         DATA 0.50890008
09B4A 499605C0 (00443)         DATA 0.57489084
09B4C 512C13C0 (00444)         DATA 0.63415766
09B4E 57E93940 (00445)         DATA 0.68600496
09B50 AE8D9640 (00446) DVLK6:  DATA -.36516072     ;DECODING TABLE FOR K6 (16)
09B52 A3C616C0 (00447)         DATA -.27940269
09B54 9834C8C0 (00448)         DATA -.18910990
09B56 8C3695C0 (00449)         DATA -.09541580
09B58 00000000 (00450)         DATA 0.00000000
09B5A 0C3695C0 (00451)         DATA 0.09541579
09B5C 1834C8C0 (00452)         DATA 0.18910989
09B5E 23C616C0 (00453)         DATA 0.27940267
09B60 2E8D9640 (00454)         DATA 0.36516071
09B62 38F80840 (00455)         DATA 0.44506935
09B64 425D2540 (00456)         DATA 0.51046758
09B66 4ADF0340 (00457)         DATA 0.58494609
09B68 527B92C0 (00458)         DATA 0.64439614
09B6A 5935F3C0 (00459)         DATA 0.69695902
09B6C 5F19B9C0 (00460)         DATA 0.74296685
09B6E 64358640 (00461)         DATA 0.78288344
                (00462) ;
                (00463) DVLG:                         ;DECODING TABLE FOR SQRT(ENERGY) (64)
09B70 5762F9BD (00464)         DATA    0.0001666767600000
09B72 60A6013D (00465)         DATA    0.0001843423110000
09B74 6AE4553D (00466)         DATA    0.0002038801790000
09B76 7638993D (00467)         DATA    0.0002254888310000
09B78 002C038E (00468)         DATA    0.0002493876930000
09B7A 0909BDBE (00469)         DATA    0.0002758195510000
09B7C 09FFF7BE (00470)         DATA    0.0003050528070000
09B7E 0B0F2FBE (00471)         DATA    0.0003373844230000
09B80 0C3A25BE (00472)         DATA    0.0003731427710000
09B82 0D85E73E (00473)         DATA    0.0004126910390000
09B84 0EF4D1BE (00474)         DATA    0.0004564309060000
09B86 10B19FBE (00475)         DATA    0.0005040066310000
09B88 124B70BE (00476)         DATA    0.0005583095120000
09B8A 143FD2BE (00477)         DATA    0.0006174830780000
09B8C 1660D13E (00478)         DATA    0.0006829281050000
```

```
09B8E 1A8FFF3E (00479)         DATA    0.00075530965800000
09B90 1D5F870E (00480)         DATA    0.00083536263300000
09B92 1E463C3E (00481)         DATA    0.00092390012300000
09B94 217FA80E (00482)         DATA    0.00102102153000000
09B96 2508248E (00483)         DATA    0.00113012122000000
09B98 28F4EA3E (00484)         DATA    0.00124989933000000
09B9A 2D4C2D8E (00485)         DATA    0.00138237237000000
09B9C 3219393E (00486)         DATA    0.00152888570000000
09B9E 3768878E (00487)         DATA    0.00169092775000000
09BA0 3D47E73E (00488)         DATA    0.00187014391000000
09BA2 43C69D8E (00489)         DATA    0.00206835470000000
09BA4 4AF58E3E (00490)         DATA    0.00228757341000000
09BA6 52E7658E (00491)         DATA    0.00253002619000000
09BA8 5B00CD3E (00492)         DATA    0.00279817614000000
09BAA 65689C3E (00493)         DATA    0.00309474609000000
09BAC 7028188E (00494)         DATA    0.00342274882000000
09BAE 7C0F348E (00495)         DATA    0.00378551555000000
09BB0 08930D3F (00496)         DATA    0.00418673054000000
09BB2 0978B33F (00497)         DATA    0.00463046966000000
09BB4 0A7D013F (00498)         DATA    0.00512123876000000
09BB6 0B99943F (00499)         DATA    0.00566402322000000
09BB8 0CD450BF (00500)         DATA    0.00626433583000000
09BBA 0E3F693F (00501)         DATA    0.00692827324000000
09BBC 0FB1663F (00502)         DATA    0.00766258029000000
09BBE 115B30BF (00503)         DATA    0.00847471342000000
09BC0 13321C3F (00504)         DATA    0.00937292306000000
09BC2 153AF13F (00505)         DATA    0.01036632990000000
09BC4 177AF9BF (00506)         DATA    0.01146502570000000
09BC6 19F80F3F (00507)         DATA    0.01268016800000000
09BC8 1CB8AABF (00508)         DATA    0.01402410050000000
09BCA 1FC3F43F (00509)         DATA    0.01551047210000000
09BCC 2321D5BF (00510)         DATA    0.01715430000000000
09BCE 26D810BF (00511)         DATA    0.01897252050000000
09BD0 2AF952BF (00512)         DATA    0.02098335950000000
09BD2 2F8751BF (00513)         DATA    0.02320732110000000
09BD4 3490E5BF (00514)         DATA    0.02566699430000000
09BD6 3A23273F (00515)         DATA    0.02838736100000000
09BD8 404C92BF (00516)         DATA    0.03139605140000000
09BDA 471D2DBF (00517)         DATA    0.03472362270000000
09BDC 4EA6B03F (00518)         DATA    0.03840387150000000
09BDE 56FCB3BF (00519)         DATA    0.04247418090000000
09BE0 6034E53F (00520)         DATA    0.04697589020000000
09BE2 6A673CBF (00521)         DATA    0.05195472200000000
09BE4 75AF3D3F (00522)         DATA    0.05746124410000000
09BE6 082273C0 (00523)         DATA    0.06355138310000000
09BE8 08FF2A40 (00524)         DATA    0.07028700040000000
09BEA 09F34540 (00525)         DATA    0.07773650530000000
09BEC 0B013F40 (00526)         DATA    0.08597556130000000
09BEE 0C28D6C0 (00527)         DATA    0.09508784860000000
09BF0 3E811240 (00528) DVLDG:  DATA 0.48977882     ;DECODING TABLE FOR SQRT(DELTA-GAIN) (4)
09BF2 65AC8C40 (00529)         DATA 0.79432824
09BF4 08F9E4C1 (00530)         DATA 1.12201844
09BF6 0BF04A41 (00531)         DATA 1.49623564
```

```
09BF8 8C0A0541 (00532) DVLU:  DATA -1.58489251    ;DECODING TABLE FOR RESIDUAL (4) (CODED 1,0,2,3)
09BFA B9C93DC0 (00533)        DATA -0.45145380
09BFC 39C93DC0 (00534)        DATA 0.45145380
09BFE 0C0A0541 (00535)        DATA 1.58489251
09C00 B9C93DC0 (00536) DVLUF: DATA -0.45145380    ;DVLU IN FOLDED BINARY ORDER (4) (CODED 0,1,2,3)
09C02 8C0A0541 (00537)        DATA -1.58489251
09C04 39C93DC0 (00538)        DATA 0.45145380
09C06 0C0A0541 (00539)        DATA 1.58489251
09C08          (00540)        END
```

```
PAGE    13:  [BBNMD]<DCA16>BBN16T.MSU.1, 10-Nov-80 19:15:02, Ed: KFIELD
             CODING AND DECODING TABLES

  CTMC1:      09800 (00013)
  CTMC2:      09810 (00022)
  CTMC3:      09D00 (00014)
  CTMDG:      099F0 (00264)
  CTMG:       09970 (00199)
  CTMK1:      09830 (00039)
  CTMK2:      09DB0 (00103)
  CTMK3:      098F0 (00135)
  CTMK4:      09910 (00151)
  CTMK5:      09930 (00167)
  CTMK6:      09950 (001D3)
  CTMU:       099F8 (00268)
  DVLC1:      09A00 (00276)
  DVLC2:      09A10 (00285)
  DVLC3:      09A80 (00277)
  DVLDG:      09DF0 (00520)
  DVLG:       09D70 (00463)
  DVLK1:      09A30 (00302)
  DVLK2:      09AB0 (00366)
  DVLK3:      09AF0 (00398)
  DVLK4:      09B10 (00414)
  DVLK5:      09D30 (00430)
  DVLK6:      09D50 (00446)
  DVLU:       09DF8 (00532)
  DVLUF:      09C00 (00536)

LINES WITH ERRORS:     0  (MAP VERSION 000101.10)   E-     0
```

TABLE OF CONTENTS

```
PAGE    1: [BBND]<DCA16>BBN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF

                    (00001) )[BBND]<DCA16>BBN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF



PAGE    2: [BBND]<DCA16>BBN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF
           BBN 16 KB/S SPEECH CODER INPUT/OUTPUT PROGRAMS

                    (00002) "BBN 16 KB/S SPEECH CODER INPUT/OUTPUT PROGRAMS
                    (00003) ;  THE IOS CODE FITS IN THE HOLE AT $4000 - $49FF.  THE CSPU CODE
                    (00004) ;    GOES IN THE HOLE AT $3800 - $3FFF.
```

```
          (00005) "SYMBOL DEFINITIONS
          (00006) ;
00000001 (00007) H$ = 1                    ;HALF-WORD ADDRESS INCREMENT
00000002 (00008) W$ = 2                    ;FULL-WORD ADDRESS INCREMENT
          (00009) ;
0000000A (00010) ACQTHR = 10               ;SYNC-SEARCH ACQUISITION THRESHOLD: NUMBER OF
          (00011) ;                        ; FRAMES OF CONSECUTIVE GOOD SYNC BITS NEEDED
          (00012) ;                        ; TO ACQUIRE SYNC.
000000E8 (00013) AFDT$ORG = $0E8           ;ARRAY FUNCTION DISPATCH TABLE ORIGIN
00007FFF (00014) BITS15 = O"77777"         ;15-BIT MASK
          (00015) ;
          (00016) ;BIT-MASKS USED IN THE PROPAR ROUTINE
00000001 (00017) B0 = 1
00000002 (00018) B1 = 2
00000004 (00019) B2 = 4
00000008 (00020) B3 = 8
00000010 (00021) B4 = $10
00000020 (00022) B5 = $20
00000003 (00023) B10 = B1+B0
00000007 (00024) B210 = B2+B1+B0
0000000C (00025) B32 = B3+B2
0000000E (00026) B321 = B3+B2+B1
00000030 (00027) B54 = B5+B4
          (00028) ;BIT-MASKS SHIFTED 1 PLACE LEFT FOR USE IN THE CORPAR ROUTINE
00000002 (00029) C0 = 2
00000004 (00030) C1 = 4
00000008 (00031) C2 = 8
00000010 (00032) C3 = $10
00000020 (00033) C4 = $20
00000040 (00034) C5 = $40
00000006 (00035) C10 = C1+C0
0000000C (00036) C21 = C2+C1
0000000E (00037) C210 = C2+C1+C0
00000018 (00038) C32 = C3+C2
0000001C (00039) C321 = C3+C2+C1
00000060 (00040) C54 = C5+C4
          (00041) ;
000021FC (00042) CSPU$NOS = $21FC          ;CSPU NO SPECIAL SUPPORT
          (00043) ;
          (00044) ;[BBND]<MAP>BBN16T-ADRS.MSO.2,  9-Sep-80 15:08:32, Ed: KFIELD
          (00045) ;SYMBOL DEFINITIONS FOR BBN16 CODING/DECODING TABLE ADDRESSES
          (00046) ; TABLES RESIDE ON BUS1, STARTING AT $9800.
          (00047) ; TABLE CONTENTS ARE DEFINED IN MODULE "BBN16T.MSO".
          (00048) ;
00009800 (00049) CTHC1=$9800               ;CODING TABLES
00009810 (00050) CTHC2=CTHC1+ 8*W$
00009800 (00051) CTHC3=CTHC1
00009830 (00052) CTHK1=CTHC2+16*W$
000098B0 (00053) CTHK2=CTHK1+64*W$
000098F0 (00054) CTHK3=CTHK2+32*W$
00009910 (00055) CTHK4=CTHK3+16*W$
00009930 (00056) CTHK5=CTHK4+16*W$
00009950 (00057) CTHK6=CTHK5+16*W$
```

SYMBOL DEFINITIONS

```
00009970 CTHG =CTHK6+16*W$
000099F0 CTHDG=CTHG +64*W$
000099F8 CTHU =CTHDG+ 4*W$
00009A00 DVLC1=CTHU + 4*W$       ;DECODING TABLES
00009A10 DVLC2=DVLC1+ 8*W$
00009A00 DVLC3=DVLC1
00009A30 DVLK1=DVLC2+16*W$
00009AB0 DVLK2=DVLK1+64*W$
00009AF0 DVLK3=DVLK2+32*W$
00009B10 DVLK4=DVLK3+16*W$
00009B30 DVLK5=DVLK4+16*W$
00009B50 DVLK6=DVLK5+16*W$
00009B70 DVLG =DVLK6+16*W$
00009BF0 DVLDG=DVLG +64*W$
00009BF8 DVLU =DVLDG+ 4*W$
00009C00 DVLUF=DVLU + 4*W$
         ;
         ;SNAP EXEC INTERRUPT ROUTINE ADDRESSES
00004022 D16$INT1 = $4022        ;MODEM SCROLL INT 1 (TMODEM)
00004030 D16$INT2 = $4030        ;MODEM SCROLL INT 2 (RMODEM)
000040FA D22$INT1 = $40FA        ;ADM INT 1
0000411E D23$INT1 = $411E        ;ADAM INT 1
         ;
000007E8 FDT$ = $7E8             ;START OF NONARRAY FUNCTION DISPATCH TABLE
         ;GFLAG SET/CLEAR CONSTANTS
00000020 SET = O"40"
00000000 CLR = 0
00000004 G0 = 4
00000005 G1 = 5
00000006 G2 = 6
00000007 G3 = 7
         ;
         ;MODEM-SCROLL INPUT DATA WORD BITS
00000080 IDW$ONHK = $80          ;LOCAL HANDSET HOOKSWITCH STATE
00000040 IDW$SI = $40            ;SIGNAL RATE INDICATOR
00000020 IDW$IC = $20            ;INCOMING CALL
00000010 IDW$CS = $10            ;CLEAR TO SEND
00000008 IDW$DM = 8              ;DATA MODE
00000004 IDW$RR = 4              ;RECEIVER READY
00000002 IDW$SQ = 2              ;SIGNAL QUALITY
00000001 IDW$RD = 1              ;RECEIVE DATA
         ;
00000502 ISVT$ = $502            ;START OF INTEGER SCALAR TABLE IN SNAP EXEC
00000004 LOSETHR = 4             ;SYNC-SEARCH LOSE-SYNC THRESHOLD: NUMBER OF
         ;                       ; SYNC ERRORS WITHOUT 2 CONSECUTIVE GOOD-SYNC
         ;                       ; FRAMES NEEDED TO LOSE SYNC.
         ;
         ;MODEM-SCROLL OUTPUT DATA WORD BITS
00000008 ODW$TR = 8              ;TERMINAL READY
00000004 ODW$RS = 4              ;REQUEST TO SEND
00000002 ODW$SR = 2              ;SIGNALING RATE
00000001 ODW$SD = 1              ;SEND DATA
         ;
```

```
PAGE    5:  [BBNU]<DCA16>BBN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF
            SYMBOL DEFINITIONS

            000000EB (00111) RANDSL = $000EB          ;LOWER AND UPPER LIMITS FOR ERRPTR
            00001C77 (00112) RANDSU = $01C77          ; IN RNISSIN
            0001FFCE (00113) SYS$FLGS = $1FFCE        ;MAP SYSTEM FLAGS REGISTER IN PSEUDO-MEMORY
            0000000C (00114) TMBITS = ODW$TR+ODW$RS   ;CONTROL BITS USED FOR TRANSMIT MODEM
                     (00115) ;
                     (00116) ;SYNCSTOP IS A "REGISTER" IN THE ADAM AND AOM USED FOR CSPU-SCROLL
                     (00117) ;  COMMUNICATION.
                     (00118)          OPADD SYNCSTOP,(16 .LS. 10) + (26 .LS. 5)+4
                     (00119) ;
                     (00120) ;THE SAF (SET ADDRESS FIELD) INSTRUCTION IS ONE OF THE NEW ONES ADDED
                     (00121) ;     TO THE CSPU BY THE "REV. 10" MICROCODE REVISION.
                     (00122)          OPADD SAF,(1 .LS. 14) + (29 .LS. 8) + $EF
                     (00123)


PAGE    6:  [BBNU]<DCA16>BBN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF
            PATCHES TO NON-ARRAY AND ARRAY FUNCTION DISPATCH TABLES

                     (00124) "PATCHES TO NON-ARRAY AND ARRAY FUNCTION DISPATCH TABLES
            000008DA (00125) .L = FDT$ + (W$ * 121)
     08DA 0000F3CCA (00126)           ADDR    PRORES          ;FCB 121 (SNAP FUNCTION PRORES)
     08DC 00003CE4 (00127)           ADDR    PROPAR          ;FCB 122 (PROPAR)
     08DE 00003E12 (00128)           ADDR    CORPAR          ;FCB 123 (CORPAR)
     08E0 0000495C (00129)           ADDR    ADAMINT         ;FCB 124 (ADINT)
     08F2 00003B00 (00130)           ADDR    TMODEMINT       ;FCB 125 (TMINT)
     08E4 00003B62 (00131)           ADDR    RMODEMINT       ;FCB 126 (RMINT)
     08F6 00003C86 (00132)           ADDR    AOMINT          ;FCB 127 (DAINT)
                     (00133) ;
                     (00134) ;
            00000A50 (00135)           .L = AFDT$ORG+3*W$*(188-128)    ;FCB 188  (DECODA)
     0A50 001F3F5A (00136)           ADDR    DCDU(R7,1)
     0A52 001E3FA0 (00137)           ADDR    DCDSA(R7,1)
     0A54 001021FC (00138)           ADDR    CSPU$NOS(,1,0)
                     (00139) ;
            00000A56 (00140)           .L = AFDT$ORG+3*W$*(189-128)    ;FCB 189  (DECODB)
     0A56 001E3F5A (00141)           ADDR    DCDU(R7,1)
     0A58 001E3FD4 (00142)           ADDR    DCDSB(R7,1)
     0A5A 001021FC (00143)           ADDR    CSPU$NOS(,1,0)
                     (00144) ;
```

```
                   (00145) "PATCHES TO INTERRUPT SERVICE ROUTINES
                   (00146) ;  THESE PATCHES MAKE SCROLL INTERRUPTS GO TO OUR OWN ROUTINES.
                   (00147) ;  NOTE THAT LOADING THIS FILE WILL MAKE THE NORMAL ADAM/ADM/IOS-2 SNAP
                   (00148) ;    FUNCTIONS INOPERATIVE!
          00004022 (00149) $L = D16$INT1
04022 E5000556 (00150) $1:      INCH     INCTR          ;COUNT INTERRUPT
04024 06003B08 (00151)          CALL     R0,TMODEMINT
04026 0F70     (00152)          RET
04027 0800     (00153)          EVEN
04028 00004022 (00154)          JMP      $1
               (00155) ;
          00004030 (00156) $L = D16$INT2
04030 E5000557 (00157) $1:      INCH     RMCTR
04032 06003B62 (00158)          CALL     R0,RMODEMINT
04034 0F70     (00159)          RET
04035 080F     (00160)          EVEN
04036 00004030 (00161)          JMP      $1
               (00162) ;
          000040FA (00163) $L = D22$INT1
040FA E5000558 (00164) $1:      INCH     DACTR
040FC 06003C86 (00165)          CALL     R0,ADMINT
040FE 0F70     (00166)          RET
040FF 0000     (00167)          EVEN
04100 000040FA (00168)          JMP      $1
               (00169) ;
          0000411E (00170) $L = D23$INT1
0411E E5000555 (00171) $1:      INCH     ADCTR
04120 0600495C (00172)          CALL     R0,ADAMINT
04122 0F70     (00173)          RET
04123 0800     (00174)          EVEN
04124 0000411E (00175)          JMP      $1
               (00176)
```

```
         "I/O BUFFER DEFINITIONS
         ; ALL BUFFERS ARE ON BUS 1.  EXCEPT AS NOTED, ALL ARE FIXED-POINT, 16
         ;  BITS,  AND HAVE SPACING OF 1 HALF-WORD.  THE SOURCE/SINK BUFFERS HAVE
         ;  ASSIGNED BID NUMBERS, BUT THE OTHERS DON'T.  THEREFORE,  IN THESE
         ;  I/O ROUTINES, WE USE THE ABSOLUTE ADDRESSES GIVEN BELOW (WHICH ARE
         ;  TAKEN FROM THE VOCODER INITIALIZATION TYPEOUT).
         ;   **** THEREFORE, IF THE BUFFER CONFIGURATIONS SHOULD CHANGE, THESE
         ;    SYMBOLS WILL NEED TO BE MODIFIED. ****
         ;
000000D8 SBLNGTH = 216            ;LENGTH OF ALL SPEECH DATA BUFFERS
00000024 SL6=SBLNGTH/6            ;1/6 LENGTH FOR A/D/A COPY ROUTINES
0000020A MBLNGTH = 522            ;LENGTH OF ALL MODEM DATA BUFFERS
00000034 ML10 = 520/10            ;1/10 LENGTH FOR TMODEM COPY ROUTINES
000000D8 NRESID = SBLNGTH         ;NUMBER OF RESIDUAL SAMPLES PER FRAME
         ;
         ;HISTOGRAM BUFFERS, WHICH ARE UPDATED BY PROPAR
         ;
00009C08 HISTO = 39944            ;(WE USE 260 OF THE 320 HALFWORDS RESERVED)
00009C08 GHIST = HISTO            ;GAIN (64)
00009C48 DGHIST = GHIST + 64      ;DELTA-GAINS (4)
00009C4C C1HIST = DGHIST + 4      ;PITCH-TAP COEFF C1 (8)
00009C54 C2HIST = C1HIST + 8      ;C2 (16)
00009C64 C3HIST = C2HIST + 16     ;C3 (8)
00009C6C K1HIST = C3HIST +8       ;K1 (64)
00009CAC K2HIST = K1HIST + 64     ;K2 (32)
00009CCC K3HIST = K2HIST + 32     ;K3 (16)
00009CDC K4HIST = K3HIST + 16     ;K4 (16)
00009CEC K5HIST = K4HIST + 16     ;K5 (16)
00009CFC K6HIST = K5HIST + 16     ;K6 (16)
00009D0C TOPHIST = K6HIST + 16
         ;
         ;VOCODER DATA BUFFERS
         ;
00009D48 TADBA = 40264            ;A/D A BUFFER
00009E20 TADBB = 40480            ;A/D B BUFFER
00009EF8 TADBC = 40696            ;A/D "ON-HOOK" TONE DATA
00009FD0 TSRA =  40912            ;TSOURCE A BUFFER
0000A0A8 TSRB =  41128            ;TSOURCE B BUFFER
0000A180 TSNKA = 41344            ;TSINK A BUFFER
0000A266 TSNKB = 41574            ;TSINK B BUFFER
0000A34C TBTA = 41804             ;TBITS A BUFFER (ALL LENGTHS = MBLNGTH)
0000A556 TBTB = 42326             ;TBITS B BUFFER
0000A760 TBTC = 42848             ;FAKE TBITS "SILENCE" DATA
0000A96A TMDMA = 43370            ;TMODEM A BUFFER
0000AB74 TMDMB = 43892            ;TMODEM B BUFFER
         ;
0000AD7E RMDMA = 44414            ;RMODEM A BUFFER
0000AF88 RMDMB = 44936            ;RMODEM B BUFFER
0000B192 RMDMC = 45458            ;RMODEM C BUFFER
0000B39C RBTA = 45980             ;RBITS A BUFFER
0000B5A6 RBTB = 46502             ;RBITS B BUFFER
0000B7B0 RSRA =  47024            ;RSOURCE A BUFFER (230 FULLWORDS)
0000B97C RSRB =  47484            ;RSOURCE B BUFFER
```

```
        I/O BUFFER DEFINITIONS

        0000BB48 (00230) RSNKA = 47944           ;RSINK A BUFFER
        0000BC20 (00231) RSNKB = 48160           ;RSINK B BUFFER
        0000BCF8 (00232) RSNKC = 48376           ;D/A "SILENCE" DATA
        0000BDD0 (00233) RDADA = 48592           ;D/A A BUFFER
        0000BEA8 (00234) RDADB = 48808           ;D/A B BUFFER
                 (00235) ;
                 (00236) ; TWO BUFFERS FOR THE SYNCSEARCH SUBROUTINE ARE DEFINED JUST BELOW THE
                 (00237) ;      16K (FULLWORD) BOUNDARY  ON BUS 1.
        0000F7BEC (00238) RSSPF = $7BEC          ;SYNC SEARCH PREVIOUS FRAME (522 HALFWORDS)
        0000F7DF6 (00239) RSSSS = $7DF6          ;SYNC SEARCH SUM-SYNC COUNTS (SAME)
                 (00240) ;
                 (00241) ;INITIALIZE ALTERNATING SYNC BITS IN TMODEM BUFFERS
        0000A96A (00242) @L = TMDMA
0A96A 000C       (00243)          DATA    0 + TMBITS
        0000AB74 (00244) @L = TMDMB
0AB74 000D       (00245)          DATA    1 + TMBITS
                 (00246)
```

```
                   (00247) 'INTEGER SCALAR TABLE DEFINITIONS FOR I/O ROUTINES
                   (00248) ;
                   (00249) ; T/R SOURCE/SINK BUFFER FLAGS.  ALL ARE INITED TO 0 AND USE THE
                   (00250) ;    CONVENTION THAT 0 = NON-FULL AND NZ = NON-EMPTY.
                   (00251) ;
          00000534 (00252) TSRFA = ISVT$+50                    ;TSOURCE FLAGS A & B
          00000535 (00253) TSRFB = ISVT$+51
          00000536 (00254) TBTFA = ISVT$+52                    ;TBITS FLAGS A & B
          00000537 (00255) TBTFB = ISVT$+53
          0000053B (00256) RBTFA = ISVT$+57                    ;RBITS FLAGS A & B
          0000053C (00257) RBTFB = ISVT$+58
          0000053D (00258) RSNFA = ISVT$+59                    ;RSINK FLAGS A & B
          0000053E (00259) RSNFB = ISVT$+60
          00000534 (00260) @L=TSRFA
00534 0000         (00261)         DATA 4D'0'
 ...
          0000053B (00262) @L=RBTFA
0053B 0000         (00263)         DATA 4D'0'
 ...
                   (00264) ;
                   (00265) ;I/O ROUTINE BUFFER/FLAG POINTER OFFSETS (AND INIT VALUES)
                   (00266) ;THESE ARE USED BY THE 4 INTERRUPT ROUTINES TO KEEP TRACK OF
                   (00267) ;  THE PROPER BUFFERS AND FLAGS.
                   (00268) ;
                   (00269) ;USED IN ADAMINT:
          00000542 (00270) ADPO =  ISVT$+64                    ;A/D OFFSET (0)
          00000543 (00271) TSRPO = ISVT$+65                    ;TSOURCE OFFSET (-2)
                   (00272) ;USED IN TMODEMINT:
          00000544 (00273) TBTPO = ISVT$+66                    ;TBITS OFFSET (0)
          00000545 (00274) TMPO =  ISVT$+67                    ;TMODEM OFFSET (0)
                   (00275) ;USED IN RMODEMINT:
          00000546 (00276) RMPO =  ISVT$+68                    ;RMODEM OFFSET (0)
          00000547 (00277) RBTPO = ISVT$+69                    ;RBITS OFFSET (-2)
                   (00278) ;USED IN ADMINT:
          00000548 (00279) RSNPO = ISVT$+70                    ;RSINK OFFSET (0)
          00000549 (00280) DAPO =  ISVT$+71                    ;D/A OFFSET (0)
          00000542 (00281) @L=ADPO
00542 0000         (00282)         DATA    0,-2,0,0,0,-2,0,0
00543 FFFE
00544 0000
00545 0000
00546 0000
00547 FFFE
00548 0000
00549 0000
                   (00283) ;
                   (00284) ; I/O BUFFER ERROR COUNTERS.  ALL ARE INITED TO 0.
                   (00285) ;
          0000054A (00286) TADDC = ISVT$+72                    ;A/D FRAME DISCARD COUNTER
          0000054B (00287) TMFFC = ISVT$+73                    ;TMODEM FAKE FRAME COUNTER
          0000054C (00288) RMFDC = ISVT$+74                    ;RMODEM FRAME DISCARD COUNTER
          0000054D (00289) RSNNR = ISVT$+75                    ;RSINK DATA NOT READY COUNTER
                   (00290) ;
```

```
                  (00291) ; MISC. INTEGER MEMORIES (AND INIT VALUES)
                  (00292) ;
         0000053F (00293) RUN =   ISVT$+61                   ;VOCODER RUN FLAG
                  (00294) ; (62 FREE FOR MPITM USE)
         0000053F (00295) $L=RUN
    053F 0001     (00296)         DATA    1                  ;FOR DEBUGGING, MUST BE "ON"
                  (00297) ;
         0000054E (00298) TFCTR = ISVT$+76                   ;TRANSMITTER FRAME COUNTER (0)
         0000054F (00299) RFCTR = ISVT$+77                   ;RECEIVER FRAME COUNTER (0)
         00000550 (00300) RLCTR = ISVT$+78                   ;RCVR LOST-SYNC COUNTER (0)
         00000551 (00301) RONHK = ISVT$+79                   ;LOCAL HANDSET ON-HOOK STATE (0)
         00000552 (00302) RSYNC = ISVT$+80                   ;STATE OF RMODEM SYNC (0)
         00000553 (00303) RBOFO = ISVT$+81                   ;BEGINNING OF FRAME OFFSET: SYNC BIT
                  (00304) ; (82 FREE FOR MPITM)              ; POSITION IN RMODEM BUFFER
         0000054A (00305) $L=TADDC
    054A 0000     (00306)         DATA    9D'0'
    ...
                  (00307) ;
                  (00308) ; INTERRUPT COUNTERS
                  (00309) ;
         00000555 (00310) ADCTR = ISVT$+83                   ;COUNTS ADAM (A/D) INTERRUPTS
         00000556 (00311) TMCTR = ISVT$+84                   ;COUNTS TMODEM INTERRUPTS
         00000557 (00312) RMCTR = ISVT$+85                   ;COUNTS RMODEM INTERRUPTS
         00000558 (00313) DACTR = ISVT$+86                   ;COUNTS AOM (D/A) INTERRUPTS
         00000555 (00314) $L=ADCTR
    0555 0000     (00315)         DATA    4D'0'
    ...
                  (00316) ;
                  (00317) ; DEBUGGING/DEMONSTRATION AIDS
                  (00318) ;
         0000057D (00319) RMCOR = ISVT$+123                  ;SET TO NZ TO TELL CORPAR NOT TO
                  (00320) ; (124 FREE FOR MPITM)             ; CORRECT CHANNEL ERRORS (0)
         0000057F (00321) RERRS = ISVT$+125                  ;SET TO N>0 TO TELL RMODEMINT TO
                  (00322) ; (126 FREE FOR MPITM)             ; SIMULATE N CHANNEL ERRORS PER FRAME(0)
         0000057D (00323) $L=RMCOR
    057D 0000     (00324)         DATA    4D'0'
    ...
                  (00325)
```

```
                        (00326) "MODEM INTERFACE AND SYSTEM CLOCKS PROGRAMS
                        (00327) ; JJW, 18-SEP-79
                        (00328) ;
                        (00329) ;THERE ARE TWO ENTRY POINTS:
                        (00330) ;  CLKSET: CLOCK RATE SETTING (START ADDRESS = 0)
                        (00331) ;    THIS ENTRY SETS THE DATA AND SAMPLE CLOCKS AND STARTS THEM.
                        (00332) ;    THE 16-BIT CLOCK RATE VALUE IS A LITERAL IN THE INSTRUCTION AT
                        (00333) ;    CLKGO.  (TRIED BINDING AN INTEGER TABLE VALUE HERE, BUT BECAUSE
                        (00334) ;    WE USE NO BIDS, LDS$ IN THE SNAP EXEC DOES NO BINDING AT ALL!)
                        (00335) ;
                        (00336) ;  RUNMOD: SET CLOCKS AND RUN MODEM TRANSMIT/RECEIVE (START ADDRESS 1)
                        (00337) ;    THIS ENTRY SETS AND STARTS THE CLOCKS AND THEN RUNS THE MODEM
                        (00338) ;    INTERFACE PROGRAM.
                        (00339) ;
                        (00340) ;THESE ENDING ADDRESSES ARE TRUNCATED TO 15 BITS BECAUSE THE LLIT
                        (00341) ;  FIELD OF THE JNE INSTRUCTION IS ONLY THAT BIG.
              00002F07  (00342) RMA$END = (RMDMA+MBLNGTH-1) .AND. BITS15
              00003191  (00343) RMB$END = (RMDMB+MBLNGTH-1) .AND. BITS15
              00003390  (00344) RMC$END = (RMDMC+MBLNGTH-1) .AND. BITS15
              00002073  (00345) TMA$END = (TMDMA+MBLNGTH-1) .AND. BITS15
              00002D70  (00346) TMB$END = (TMDMB+MBLNGTH-1) .AND. BITS15
                        (00347) ;
                        (00348) ;CONSTANT FOR SETTING CLOCK RATES TO MODEM=16000 BPS, SIGNAL
                        (00349) ;  SAMPLE RATE=6621 SAMPLES/SEC.
              0000F4C6  (00350) CLKRATES = $F4C6
                        (00351) ;
                        (00352) ;REGISTER AND FLAG USAGE:
                        (00353) ;  R0 - RCVR BUFFER SWITCH, SET TO MOD$RA, MOD$RB, OR MOD$RC
                        (00354) ;  R1 - RCVR ADDRESS POINTER
                        (00355) ;  R2 - XMTR ADDRESS POINTER
                        (00356) ;  P1 SET SIGNIFIES RCVR DATUM READY
                        (00357) ;  P2 SET SIGNIFIES XMTR READY FOR NEXT DATUM
                        (00358) ;  F1 IS USED AS XMTR BUFFER SWITCH: CLEAR=>TMODEMA, SET=>TMODEMB
                        (00359) ;  F2 IS USED TO REMEMBER WHICH START ADDRESS WAS USED.
                        (00360) ;  INT1 IS USED TO SIGNAL END OF TRANSMITTER BUFFER
                        (00361) ;  INT2 IS USED TO SIGNAL END OF RECEIVER BUFFER
                        (00362) ;
              00004800  (00363) #L = $4800                    ;BUS 1 ADR FOR RTMODEM IOS MODULE
              00000000  (00364) #A = 0
                        (00365)        EVEN
     04800 006E         (00366)        DATA    MOD$SZ,0              ;SCROLL PGM SIZE IN HW, BIDS 1:0
     04801 0000
                        (00367) ;
                        (00368) MOD$BGN: BEGIN  IOS2(RTMODEM)
                        (00369) ;
 A00 04802 02300300     (00370) CLKSET: SF2,JUMP(CLKGO)              ;ENTRY FOR SETTING CLOCKS ONLY
 A01 04804 02300300     (00371) RUNMOD: CF2                          ;ENTRY FOR CLOCKS AND MODEM
 A02 04806 0300F4C6     (00372) CLKGO:  LOAD(R0,CLKRATES)            ;SET DATA AND SAMPLE CLOCKS
 A03 04808 04391800     (00373)        ADDL(R0,0,TP),JIFC(MOD$INIT,F2)
 A05 0480C 06306300     (00374)        STOP, CF7
                        (00375) ;
 A06 0480E 07000015     (00376) MOD$INIT: LOAD(R0,MOD$RA)           ;INIT RCVR BUFFER SWITCH
 A07 04810 0042AD7D     (00377)        LOAD(R1,RMDMA-1(1),L)        ;INIT RCVR PTR
```

```
A08 04812 093A0580 (00378)           CF1                             ;INIT XMTR BUFFER SWITCH
A09 04814 0A02A969 (00379)           LOAD(R2,TMDMA-1(1),L)           ;INIT XMTR PTR
A0A 04816 0C340C00 (00380) MOD$LOOP: JIFS(MOD$RCVR,P1)               ;P1=>RCVR HAS DATUM
A0B 04818 00000000
A0D 0481C 3E341C00 (00381)           JIFS(MOD$XMTR,P2)               ;P2=>XMTR READY FOR NEXT DATUM
A0F 04820 0A300000 (00382)           JUMP(MOD$LOOP)
                   (00383) ;
                   (00384) ;  MODEM RECEIVER
                   (00385) ;
A10 04822 11300600 (00386) MOD$RCVR: MW                              ;SET DIRECTION MODEM-TO-MAP
A11 04824 122C001C (00387)           JEQ(MOD$RB,R0,MOD$RB)           ;DISPATCH TO PROPER BUFFER
A13 04828 142C0022 (00388)           JEQ(MOD$RC,R0,MOD$RC)           ; FILLER-UPPER
A15 0482C 165A0001 (00389) MOD$RA:   ADD(R1,1,TM)                    ;INCR RPTR AND READ DATUM
A16 0482E 10602F07 (00390)           JNE(MOD$LOOP,R1,RMA$END)        ;FILLED RMODEMA BUFFER?
A17 04830 00000000
A19 04834 1A00001C (00391)           LOAD(R0,MOD$RB)                 ;YES, SET SWITCH TO B
A1A 04836 1042AF07 (00392)           LOAD(R1,RMDMB-1(1),L)           ; AND RPTR TO RMODEMB
A1B 04838 0A304000 (00393)           INT2, JUMP(MOD$LOOP)            ; AND INT LINE 2 TO SAY FULL
                   (00394) ;
A1C 0483A 105A0001 (00395) MOD$RB:   ADD(R1,1,TM)                    ;SAME FOR RMODEMB
A1D 0483C 1E683191 (00396)           JNE(MOD$LOOP,R1,RMB$END)
A1F 04840 20000022 (00397)           LOAD(R0,MOD$RC)
A20 04842 2147B191 (00398)           LOAD(R1,RMDMC-1(1),L)
A21 04844 0A3A4000 (00399)           INT2, JUMP(MOD$LOOP)
                   (00400) ;
A22 04846 235A0001 (00401) MOD$RC:   ADD(R1,1,TM)                    ;AND FOR RMODEMC
A23 0484B 246A339A (00402)           JNE(MOD$LOOP,R1,RMC$END)
A25 0484C 26000015 (00403)           LOAD(R0,MOD$RA)
A26 0484E 2742A070 (00404)           LOAD(R1,RMDMA-1(1),L)
A27 04850 0A304000 (00405)           INT2, JUMP(MOD$LOOP)
                   (00406) ;
                   (00407) ;  MODEM TRANSMITTER
                   (00408) ;
A28 04852 2A340A80 (00409) MOD$XMTR: MR, JIFS(MOD$TB,F1)             ;SET DIRECTION MAP-TO-MODEM
A29 04854 00000000
                   (00410) ;                                         ;F1 SET MEANS USE TMODEMB
A2B 04858 2C9A0001 (00411) MOD$TA:   ADD(R2,1,TM)                    ;SEND WORD FROM TMODEMA
A2C 0485A 2EA02073 (00412)           JNE(MOD$LOOP,R2,TMA$END)        ;EMPTIED TMODEMA?
A2D 0485C 00000000
A2F 04860 3082AB73 (00413)           LOAD(R2,TMDMB-1(1),L)           ;YES, RESET TPTR, FLIP F1,
A30 04862 0A302100 (00414)           SF1, INT1, JUMP(MOD$LOOP)       ; INT1 TO SAY BUFFER EMPTY
                   (00415) ;
A31 04864 329A0001 (00416) MOD$TB:   ADD(R2,1,TM)                    ;SAME FOR TMODEMB
A32 04866 34A8207D (00417)           JNE(MOD$LOOP,R2,TMB$END)
A33 04868 00000000
A35 0486C 3602A969 (00418)           LOAD(R2,TMDMA-1(1),L)
A36 0486E 0A302180 (00419)           CF1, INT1, JUMP(MOD$LOOP)
                   (00420) ;
    04870          (00421)           END
                   (00422) ;
        0000006E   (00423) MOD$SZ = $L-MOD$BGN               ;SIZE OF SCROLL PROGRAM IN HW
    04870 0000     (00424)           DATA    0               ;DIR=FROM MAP (PRIOR MEANINGLESS)
    04871 0001     (00425)           DATA    $0001           ;$CHNLS=0 : AQ. MODE = DOUBLE (ALSO)
```

```
04072 0000     (00426)        DATA    0               ;BID3 : BID2
04073 0000     (00427)        DATA    0,0             ;NO CONTROL REGISTERS
04074 0000
04075 FFFF     (00428)        DATA    -1,-1,-1,-1,-1  ;NULL OFFSET, BUFFER CHAIN ANCHORS
04076 FFFF
04077 FFFF
04078 FFFF
04079 FFFF
               (00429)
```

```
                        (00430) 'ADAM SINGLE-CHANNEL DOUBLE-BUFFERED SAMPLING
                        (00431) ; JJW, 20 SEPT 79
                        (00432) ;
                        (00433) ; WE SAMPLE ON CHANNEL 1, WHICH IS CONNECTED TO THE GTE SPI.
                        (00434) ; THIS PROGRAM IS FUNCTIONALLY IDENTICAL TO ADMSD, BUT BOUND AT
                        (00435) ;    ASSEMBLY TIME.
                        (00436) ;
               00001E1F (00437) ADA$END = (TADBA+SBLNGTH-1) .AND. BITS15
               00001EF7 (00438) ADB$END = (TADBB+SBLNGTH-1) .AND. BITS15
                        (00439) ;
               00000002 (00440) ADCNTRL = 2              ;INTERNAL TRIGGER, EXTERNAL SAMPLING CLOCK,
                        (00441) ;                        ; CH1 CONTROLLED BY SR1, SHORT FLT PT
                        (00442) ;
                        (00443) ; REGISTER AND FLAG USAGE:
                        (00444) ;    R0 - CONTROLS MUX
                        (00445) ;    R1 - DATA ADDRESS POINTER
                        (00446) ;    F1 - THE ADAM USES F1 TO TURN SAMPLING ON/OFF
                        (00447) ;    INT1 - SIGNALS END OF BUFFER
                        (00448) ;
               000040A0 (00449) .L = $40A0                          ;BUS 1 ADR FOR ADPROG IOS MODULE
               00000000 (00450) .A = 0
                        (00451)         EVEN
     040A0 002C         (00452)         DATA AD$SZ,0                ;SCROLL PGN SIZE (HW), BIDS 1:0
     040A1 0000
                        (00453) ;
                        (00454) AD$BGN: BEGIN IOS2(ADPROG)
                        (00455) ;
A00  040A2 01300600     (00456)         PR                          ;SET DIRECTION ADAM-TO-MAP
A01  040A4 02000001     (00457)         LOAD(R0,$0001)              ;INIT MUX TO CHNL 1
A02  040A6 03190000     (00458)         ADD(R0,0,TP)
A03  040A8 04300100     (00459)         SF1                         ;START SAMPLING
A04  040AA 05429D47     (00460) AD$LOOP: LOAD(R1,TADBA-1(1),L)  ;INIT RPTR TO A/D BUFFER A
A05  040AC 06190010     (00461) .1:     ADD(R0,$10,TP)              ;RELEASE S/H
A06  040AE 07100010     (00462)         SUB(R0,$10)
A07  040B0 085A0001     (00463)         ADD(R1,1,TM)                ;READ SAMPLE, SEND TO BUFFER
A08  040B2 0A6A1E1F     (00464)         JNE(.1,R1,ADA$END)          ;FILLED BUFFER?
A09  040B4 00000009
A0A  040B8 0C343000     (00465)         JIFS(AD$STOP,SYNCSTOP),INT1  ;YES:  INT1 TO SAY FULL, AND CHECK
                        (00466) ;                                    ;  IF CSPU SAYS TO STOP
A0D  040BC 0F429E1F     (00467)         LOAD(R1,TADBB-1(1),L)  ;INIT RPTR TO A/D BUFFER B
A0E  040BE 0F190010     (00468) .2:     ADD(R0,$10,TP)
A0F  040C0 10100010     (00469)         SUB(R0,$10)
A10  040C2 115A0001     (00470)         ADD(R1,1,TM)
A11  040C4 126A1EF7     (00471)         JNE(.2,R1,ADB$END)
A13  040C8 14303000     (00472)         JIFC(AD$LOOP,SYNCSTOP), INT1
A15  040CC 16306100     (00473) AD$STOP: CF1, STOP              ;STOP SAMPLING AND HALT
     040CE              (00474)         END
                        (00475) ;
               0000002C (00476) AD$SZ = .L-AD$BGN
     040CE 0002         (00477)         DATA 2                      ;DIR = TO MAP
     040CF 0101         (00478)         DATA $0101                  ;#CHNLS=1 : AQ. MODE = DOUBLE
     040D0 0000         (00479)         DATA 0                      ;BID3 : BID2
     040D1 0003         (00480)         DATA 3,0                    ;LOAD 3 CONTROL REGS STARTING WITH 0
```

```
PAGE   16:  [BBND]<DCA16>MBN16U.MSU.2, 29-Dec-80 15:44:23, Ed: WOLF
            ADAM SINGLE-CHANNEL DOUBLE-BUFFERED SAMPLING

      04DD2 0000
      04DD3 0000     (00401)        DATA 0                   ;REG. 0: SAMPLE RATE 1 (FOR EXT CLK)
      04DD4 0000     (00402)        DATA 0                   ;REG. 1: SAMPLE RATE 2 NOT USED
      04DD5 0002     (00403)        DATA ADCNTRL             ;REG. 2: ADAM CONTROL BITS
      04DD6 FFFF     (00404)        DATA -1,-1,-1,-1,-1      ;NULL OFFSET, BUFFER CHAIN ANCHORS
      04DD7 FFFF
      04DD8 FFFF
      04DD9 FFFF
      04DDA FFFF
                     (00405)
```

```
PAGE   17:  [BBND]<DCA16>RAN160.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF
            ADM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT

                          (00486) "ADM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT
                          (00487) ; JJW, 20 SEPT 79
                          (00488) ;
                          (00489) ;D/A CHANNEL 0 OUTPUTS AN INTERNALLY-GENERATED RAMP, WHILE CHANNEL 1
                          (00490) ;  (CONNECTED TO THE GTE SPI) OUTPUTS DOUBLE-BUFFERED DATA FROM BUS 1.
                          (00491) ;  THIS PROGRAM IS FUNCTIONALLY SIMILAR TO ADMID, BUT BOUND AT ASSEMBLY
                          (00492) ;  TIME.
                          (00493) ;
                 00003EA7 (00494) DAA$END = (RDABA+SBLNGTH-1) .AND. BITS15
                 00003F7F (00495) DAB$END = (RDABB+SBLNGTH-1) .AND. BITS15
                          (00496) ;
                 00000002 (00497) DACNTRL = 0+2+0          ;SINGLE CHANNEL MODE, EXTERNAL CLOCK,
                          (00498) ;                        ; SHORT FLT PT
                          (00499) ; REGISTER AND FLAG USAGE:
                          (00500) ;   R0 - HOLDS RAMP VALUE FOR CH0 DAC
                          (00501) ;   R1 - DATA ADDRESS POINTER FOR CH1 DAC
                          (00502) ;   F1 - CAN BE USED FOR SCOPE SYNC
                          (00503) ;   INT1 - SIGNALS END OF BUFFER
                          (00504) ;
                 000048E0 (00505) #L = $48E0                         ;BUS 1 ADR FOR DAPROG IOS MODULE
                 00000000 (00506) #A = 0
                          (00507)         EVEN
     048E0 002C           (00508)         DATA DA$SZ,0               ;SCROLL PGM SIZE (HW), BIDS 1:0
     048E1 0000
                          (00509) ;
                          (00510) DA$BGN: BEGIN IOS2(DAPROG)
 A00 048E2 01300600 (00511)         PW                          ;SET DIRECTION MAP-TO-ADM
 A01 048E4 02000000 (00512) DA$LOOP: LOAD(R0,0)                 ;INITIALIZE CH0 RAMP
 A02 048E6 03428DCF (00513)         LOAD(R1,RDABA-1(1),L)       ;INIT TPTR TO D/A BUFFER A
 A03 048E8 04300100 (00514)         SF1                         ;F1 CAN BE USED FOR SCOPE SYNC
 A04 048EA 0519000A (00515) #1:     ADD(R0,10,TP)               ;OUTPUT RAMP TO CH0 DAC
 A05 048EC 065A0001 (00516)         ADD(R1,1,TM)                ;GET SAMPLE FOR CH1 DAC
 A06 048EE 08683EA7 (00517)         JNE(#1,R1,DAA$END)          ;EMPTIED BUFFER?
 A07 048F0 00000000
 A09 048F4 0A343180 (00518)         JIFS(DA$STOP,SYNCSTOP),INT1,CF1  ;YES:  INT1 TO SAY EMPTY, AND
                          (00519) ;                                    ; CHECK IF CSPU SAYS TO STOP
 A0B 048F8 0C000000 (00520)         LOAD(R0,0)                  ;REINIT CH0 RAMP
 A0C 048FA 0D428EA7 (00521)         LOAD(R1,RDABB-1(1),L)       ;INIT TPTR TO D/A BUFFER B
 A0D 048FC 0E300100 (00522)         SF1
 A0E 048FE 0F19000A (00523) #2:     ADD(R0,10,TP)
 A0F 04900 105A0001 (00524)         ADD(R1,1,TM)
 A10 04902 12683F7F (00525)         JNE(#2,R1,DAB$END)
 A11 04904 00000000
 A13 04908 14303180 (00526)         JIFC(DA$LOOP,SYNCSTOP), INT1, CF1
 A15 0490C 16306000 (00527) DA$STOP: STOP
     0490E          (00528)         END
                          (00529) ;
                 0000002C (00530) DA$SZ = #L-DA$BGN
     0490E 0000           (00531)         DATA 0                     ;DIR = FROM MAP
     0490F 0001           (00532)         DATA $0001                 ;#CHNLS=NULL : AQ. MODE = DOUBLE
     04910 0000           (00533)         DATA 0                     ;BID3 : BID2
     04911 0003           (00534)         DATA 3,0                   ;LOAD 3 CONTROL REGS STARTING WITH 0
     04912 0000
```

AOM SINGLE-CHANNEL DOUBLE-BUFFERED D/A OUT

```
04913 0000      (00535)        DATA 0                    ;REG. 0: SAMPLE RATE
04914 0000      (00536)        DATA 0                    ;REG. 1 NOT USED
04915 0002      (00537)        DATA DACNTRL              ;REG. 2: AOM CONTROL BITS
04916 FFFF      (00538)        DATA -1,-1,-1,-1,-1       ;NULL OFFSET, BUFFER CHAIN ANCHORS
04917 FFFF
04918 FFFF
04919 FFFF
0491A FFFF
                (00539)
                (00540)
```

```
                            "ADAMINT: ADAM INTERRUPT SERVICE ROUTINE
                            ; JJW, 3-OCT-79
                            ;
                            ;ADAMINT COPIES A/D DATA FROM AN A/D BUFFER TO A TSOURCE BUFFER.  IF
                            ;  THE LOCAL HANDSET IS ON-HOOK, IT COPIES A LOW-AMPLITUDE "ON-HOOK"
                            ;  TONE FROM TADBC INSTEAD, TO CONVEY THIS FACT TO THE REMOTE LISTENER.
                            ;  IF THE TSOURCE BUFFER FLAG SHOWS THE BUFFER ISN'T YET AVAILABLE, THE
                            ;  FRAME OF A/D DATA IS SIMPLY DISCARDED.
                            ;
               00004950     ΦL = $4950                    ;BUS 1 ADR FOR ADAMINT CSPU MODULE
                            ;POINTERS TO A/D BUFFERS AND TSOURCE FLAGS AND BUFFER-COPYING ROUTINES.
                            ;THESE BUFFERS AND FLAGS ARE ACCESSED BY MEANS OF POINTER OFFSETS,
                            ;  WHICH ARE IN THE INTEGER SCALAR TABLE.
                            ;       ADPO IS THE A/D BUFFER POINTER OFFSET (-2,0; INIT TO 0 SO THAT
                            ;          IT GETS SET TO TADBA ON THE FIRST CALL)
                            ;       TSRPO IS THE TSOURCE BUFFER/FLAG POINTER OFFSET (-2,0)
                            ;         INIT TO -2 SO THAT IT POINTS TO TSOURCEA ON THE 1ST CALL)
                            ;
                                    EVEN
04950 0000F9D46                     ADDR     TADBA-2          ;A/D BUFFER A (-2 FOR THE BNOVEL)
04952 00009E1E              ADBPTR: ADDR     TADBB-2          ;A/D BUFFER B
                            ;
04954 0000498A                      ADDR     AD$CPYA          ;ROUTINE TO COPY TO TSOURCE A BUFFER
04956 00004994              TSRBPTR: ADDR    AD$CPYB          ;ROUTINE TO COPY TO TSOURCE B BUFFER
                            ;
04958 00000534                      ADDR     TSRFA            ;TSOURCE FLAG A
0495A 00000535              TSRFPTR: ADDR    TSRFB            ;TSOURCE FLAG B
                            ;
                                    EVEN
0495C F04DFFCE              ADAMINT: MOVLM   SET+G2,SYS$FLGS  ;SET G2 TO MARK START OF ADAMINT
0495E F0100542                      MOVMR    R1,ADPO          ;R1 GETS A/D BUFF PTR OFFSET
04960 9411FFFE                      XORIR    R1,-2            ;SWITCH A/D BUFFERS (0,-2,0,...)
04962 E0100542                      MOVRM    R1,ADPO
04964 E200053F                      CMPMZ    RUN              ;IS THE VOCODER RUNNING?
04966 00104982                      JMP      AD$RTN,EQZ       ;NO, SO DO NIL
04968 F0300543                      MOVMR    R3,TSRPO         ;R3 GETS TSOURCE OFFSET
0496A E206495A                      CMPMZ    @TSRFPTR(R3)     ;IS THE TSOURCE BUFFER EMPTY?
0496C 01104986                      JMP      AD$DISC,NEZ      ;NO: DISCARD A/D DATA
0496E 90A24952                      MOVIR    R2,@ADBPTR(R1)   ;YES: GET A/D BUFFER-1 PTR
04970 E2000551                      CMPMZ    RONHK            ;IS LOCAL HANDSET ON HOOK?
04972 1A10                          SKPL     EQZ
04973 90209EF6                      MOVIR    R2,TADBC-2       ;YES: XMIT "ON-HOOK" TONE INSTEAD OF A/D
04975 0A00                          EVEN
04976 86064956                      CALL     R0,@TSRBPTR(R3)  ;COPY FROM A/D BUFFER TO TSOURCE BUFFER
04978 90100001                      MOVIR    R1,1
0497A E096495A                      MOVRM    R1,@TSRFPTR(R3)  ;SET TSOURCE FLAG TO SHOW FULL
0497C 9431FFFE                      XORIR    R3,-2            ;SWITCH TSOURCE BUFFERS (-2,0,-2,...)
0497E E0300543                      MOVRM    R3,TSRPO
04980 E500054E                      INCM     TFCTR            ;COUNT FRAME (TO BE) TRANSMITTED
04982 F00DFFCE              AD$RTN: MOVLM    CLR+G2,SYS$FLGS  ;CLEAR G2 TO MARK END OF ADAMINT
04984 0E70                          RETURN
                            ;
                            ; TSOURCE BUFFER ISN'T AVAILABLE, SO WE DISCARD THIS BUFFER OF NEW A/D
```

```
                    (00594) ;   DATA.  JUST COUNT THIS FRAME AS DISCARDED.
04985 0000          (00595)         EVEN
04986 E500054A      (00596) AD$DISC: INCM   TADDC
04988 2107          (00597)         NOP     AD$RTN
                    (00598) ;
                    (00599) ;ROUTINES TO COPY FROM THE A/D BUFFER POINTED TO BY (R2)+1
                    (00600) ;  TO A TSOURCE BUFFER.  A SINGLE BMOVE WOULD SUFFICE, BUT THAT WOULD
                    (00601) ;  HOLD OFF INTERRUPTS FOR ABOUT 250 USEC, SO WE BREAK IT UP INTO A FEW,
                    (00602) ;  WHICH IS ALMOST AS FAST, BUT MAKES FOR BETTER INTERRUPT LATENCY.
                    (00603) ;THE REV-18 CSPU MICROCODE REVISION MODIFIED THE BMOVE R,X,MEM SO THAT
                    (00604) ;  (1) R IS LEFT POINTING TO THE LAST SOURCE-WORD MOVED, AND
                    (00605) ;  (2) X IS LEFT WITH ITS ORIGINAL CONTENTS.  THIS LETS US USE
                    (00606) ;  THE BMOVES AS DONE BELOW WITHOUT RESETTING R AND X BEFORE EACH ONE.
04989 0000          (00607)         EVEN
0498A 90400023      (00608) AD$CPYA: MOVIR  R4,SL6-1
0498C D7289FD0      (00609)         BMOVEL  R2,R4,TSRA
0498E D728A010      (00610)         BMOVEL  R2,R4,TSRA+2*SL6
04990 D728A060      (00611)         BMOVEL  R2,R4,TSRA+4*SL6
04992 0E70          (00612)         RETURN
                    (00613)
04993 0000          (00614)         EVEN
04994 90400023      (00615) AD$CPYB: MOVIR  R4,SL6-1
04996 D728A0A0      (00616)         BMOVEL  R2,R4,TSRB
04998 D728A0F0      (00617)         BMOVEL  R2,R4,TSRB+2*SL6
0499A D728A130      (00618)         BMOVEL  R2,R4,TSRB+4*SL6
0499C 0E70          (00619)         RETURN
                    (00620)
```

```
                   "TMODEMINT: TMODEM INTERRUPT SERVICE ROUTINE
                   ; JJW, 3-OCT-79; MOD FOR DCA16, 11-SEP-80
                   ;
                   ;TMODEMINT COPIES PROTECTED AND BITSTREAMED DATA FROM A TBITS
                   ;  BUFFER TO A TMODEM BUFFER.  IF THE TBITS BUFFER FLAG SHOWS THAT
                   ;  THE BUFFER ISN'T READY YET, A BUFFER OF "FAKE" DATA IS COPIED
                   ;  FROM TBTC INSTEAD.
                   ;
         00003B00  .L = $3B00                ;PUT THE REST OF THE CSPU CODE IN $3B00-$3FFF.
                   ;
                   ; POINTER OFFSETS:
                   ;   TBTPO IS FOR TBITS BUFFERS AND FLAGS (-2,0; INIT TO 0 SO IT
                   ;     LOOKS AT TBTB THE FIRST TIME)
                   ;   TMPO IS FOR TMODEM BUFFERS (-2,0; INIT TO 0 SO IT GETS SWITCHED
                   ;     TO TMDMA THE FIRST TIME)
                   ;
                   ;POINTERS TO THE TBITS BUFFERS/FLAGS ARE FOUND IN THE PROTECT MODULE.
                   ;POINTERS TO TMODEM-COPYING ROUTINES:
                             EVEN
03B00 00000536               ADDR    TBTFA             ;TBITS BUFFER FLAGS
03B02 00000537     TBTFPTR:  ADDR    TBTFB
                   ;
03B04 00003B32               ADDR    TMI$CPYA          ;ROUTINE TO COPY INTO TMODEMA
03B06 00003B44     TMPTR:    ADDR    TMI$CPYB          ;ROUTINE TO COPY INTO TMODEMB
                   ;
                             EVEN
                   TMODEMINT:
03B08 F84DFFCE               MOVLM   SET+G2,SYS$FLGS  ;MARK START OF TMODEMINT
03B0A F0200545               MOVMR   R2,TMPO           ;R2 GETS TMODEM BUFFER PTR OFFSET
03B0C 9421FFFE               XORIR   R2,-2             ;SWITCH TO NEXT TMODEM BUFFER
03B0E E0200545               MOVRM   R2,TMPO
03B10 E200053F               CMPMZ   RUN
03B12 88103B26               JMP     TMI$RTN,EQZ       ;DO NIL IF VOCODER STOPPED
03B14 F0300544               MOVMR   R3,TBTPO          ;R3 GETS TBITS PTR OFFSET
03B16 E2063B02               CMPMZ   @TBTFPTR(R3)      ;IS THE TBITS BUFFER READY (FULL)?
03B18 88103B2A               JMP     TMI$FAKE,EQZ      ;NO, SO WE RUN FAKE DATA TO THE TMODEM
03B1A 98963CC0               MOVIR   R1,@TBTBPTR(R3)  ;YES: R1 POINTS TO TBITS BUFFER+1
03B1C 9E100001               SUBIR   R1,1              ;MAKE IT POINT TO 1ST WORD
03B1E 86843B06               CALL    R0,@TMPTR(R2)    ;COPY FROM TBITS TO TMODEM BUFFER
03B20 CC063B02               MOVZM   @TBTFPTR(R3)     ;CLEAR TBITS FLAG TO SHOW EMPTY
03B22 9431FFFE               XORIR   R3,-2             ;SWITCH TBITS BUFFERS (-2,0,-2,...)
03B24 E0300544               MOVRM   R3,TBTPO
03B26 F80DFFCE     TMI$RTN:  MOVLM   CLR+G2,SYS$FLGS  ;MARK END OF TMODEMINT
03B28 0E70                   RETURN
                   ;
                   ;TBITS(X) ISN'T FULL YET, AND WE HAVE TO SEND SOMETHING TO THE
                   ;  MODEM, SO WE HAVE HANDY A BUFFER (TBTC) OF FAKE TBITS DATA
                   ;  (CORRESPONDING TO SILENCE), WHICH WE SIMPLY USE INSTEAD.
03B29 0000                   EVEN
03B2A E500054A     TMI$FAKE: INCM    TMFFC             ;INCR TMODEM FAKE FRAME COUNT
03B2C 9010A760               MOVIR   R1,TBTC
03B2E 86843B06               CALL    R0,@TMPTR(R2)
03B30 2100                   NOP     TMI$RTN           ;RETURN WITHOUT SWITCHING TBITS
```

```
                    ;                                   ; BUFFER POINTERS
                    ;
                    ;ROUTINES FOR COPYING FROM THE TBITS BUFFER TO THE TMODEM BUFFER.
                    ;  R1 COMES POINTING TO THE 1ST WORD OF THE TBITS BUFFER (THE STREAM
                    ;  STARTS IN THE 2ND WORD, BUT WE WANT THE 3RD WORD-2 FOR THE BMOVEL).
                    ;WE COPY THE 2ND THRU THE 522ND HALFWORD, OMITTING THE (INITIAL)
                    ;  SYNC-BIT WORD.
                    ;THE TBITS BUFFER HAS THESE DATA IN EXACTLY THE SAME POSITIONS (RELATIVE
                    ;  TO THE START OF THE BUFFER) AS IN THE TMODEM BUFFER.
03B31 0000                  EVEN
03B32 F0420001      TMI$CPYA: MOVHR R4,1(R1)            ;MOVE 2ND HALFWORD
03B34 E040A96D              MOVRM   R4,TMDMA+1          ;INTO HALFWORD AFTER SYNC BIT
03B36 90400033              MOVIR   R4,ML10-1
03B38 D710A96C              BMOVEL  R1,R4,TMDMA+2   ;MOVE THE REST IN 5 HUNKS OF 52 FW
03B3A D710A9D4              BMOVEL  R1,R4,TMDMA+2+2*ML10
03B3C D710AA3C              BMOVEL  R1,R4,TMDMA+2+4*ML10
03B3E D710AAA4              BMOVEL  R1,R4,TMDMA+2+6*ML10
03B40 D710AB0C              BMOVEL  R1,R4,TMDMA+2+8*ML10
03B42 0E70                  RETURN
                    ;
03B43 0000                  EVEN
03B44 F0420001      TMI$CPYB: MOVHR R4,1(R1)
03B46 E040AD75              MOVRM   R4,TMDMB+1
03B48 90400033              MOVIR   R4,ML10-1
03B4A D710AD76              BMOVEL  R1,R4,TMDMB+2
03B4C D710ADDE              BMOVEL  R1,R4,TMDMB+2+2*ML10
03B4E D710AC46              BMOVEL  R1,R4,TMDMB+2+4*ML10
03B50 D710ACAE              BMOVEL  R1,R4,TMDMB+2+6*ML10
03B52 D710AD16              BMOVEL  R1,R4,TMDMB+2+8*ML10
03B54 0E70                  RETURN
```

```
                'RMODEMINT: RMODEM INTERRUPT SERVICE MODULE
                ; JJW, 25 SEPT 79
                ;
                ;RMODEMINT CHECKS SYNC ON THE BUFFER JUST INPUT BY THE RMODEM SCROLL
                ;  PROGRAM, AND IF SYNC IS GOOD, COPIES THE LATEST "FRAME" OF DATA
                ;  FROM THE RMODEM BUFFERS TO A RBITS BUFFER.  A "FRAME" OF DATA IS
                ;  MBLNGTH WORDS STARTING WITH A SYNC-BIT WORD; IN GENERAL, IT DOES
                ;  NOT START AND END ON RMODEM BUFFER BOUNDARIES.
                ;THE STATE OF FRAME SYNC IS GIVEN BY RSYNC:
                ;  RSYNC=0 MEANS WE KNOW NOTHING ABOUT SYNC, SO WE MUST CALL SYNCINIT
                ;    TO INITIALIZE SYNC-SEARCHING.
                ;  RSYNC=-1 MEANS WE ARE STILL IN THE PROCESS ON SCANNING INCOMING
                ;    BUFFERS FOR FRAME SYNC (SYNCSRCH).
                ;  RSYNC=+1 MEANS WE HAVE SYNC, SO WE CALL SYNCUPDATE TO CHECK THE NEW
                ;    FRAME FOR CONTINUED SYNC.  IF SYNCUPDATE SETS RSYNC TO 0, THEN WE
                ;    HAVE LOST SYNC AND WE MUST REINITIALIZE WITH SYNCINIT.
                ;RMODEMINT ALSO:
                ;  COPIES THE ON-HOOK BIT OF THE 1ST DATUM IN EACH RMODEM
                ;    BUFFER INTO RONHK FOR USE BY THE ADAMINT MODULE,
                ;  SIMULATES CHANNEL ERRORS IF RERRS IS NONZERO, AND
                ;  CHECKS TO BE SURE THAT THE RBITS BUFFER IS AVAILABLE; IF NOT, THEN
                ;    THE NEW FRAME OF RMODEM DATA IS SIMPLY DISCARDED.
                ;
                ;POINTER OFFSETS:
                ;  RBTPO IS FOR THE RBITS BUFFERS AND FLAGS (-2,0; INIT TO -2
                ;    SO THAT IT POINTS TO RBTBA THE FIRST TIME)
                ;  RMPO IS FOR RMODEM BUFFERS (-4,-2,0; INIT TO 0 SO IT GETS SWITCHED
                ;    TO RMDMA THE FIRST TIME)
                ;
                ;POINTERS TO RMODEM BUFFERS AND RBITS FLAGS.  (POINTERS TO
                ;  RBITS BUFFERS ARE IN THE CORPAR MODULE.)
                ;USE RBTBPTR(RBTPO) TO POINT TO THE CURRENT RBITS BUFFER AND
                ;  RBTFPTR(RBTPO) TO POINT TO THE CURRENT FLAG.
03B55 0000                    EVEN
03B56 0000053B                ADDR     RBTFA
03B58 0000053C      RBTFPTR:  ADDR     RBTFB
                ;
03B5A 00000192                ADDR     RMDMC     ;USE RMPTR(RMPO) TO POINT TO THE
03B5C 0000AD7E                ADDR     RMDMA     ; CURRENT RMODEM BUFFER, OR RMPTR-2(RMPO)
03B5E 0000AF00                ADDR     RMDMB     ; TO POINT TO THE PREVIOUS ONE
03B60 00000192      RMPTR:    ADDR     RMDMC
                ;
                ;  NOTE THAT RMODEMINT IS TO BE ENTERED BY CALL R0,RMODEMINT,
                ;   SO IT SIMPLY RETURNS.  THERE WILL BE ANOTHER LEVEL THAT
                ;   IS DISPATCHED TO BY THE INTERRUPT, SUCH AS:
                ;             CALL     R0,RMODEMINT
                ;             RET
                ;
                              EVEN
                    RMODEMINT:
03B62 F04DFFCF                MOVLM    SET+G2,SYS$FLGS  ;MARK START OF RMODEMINT
03B64 F0200546                MOVMR    R2,RMPO          ;R2 GETS RMODEM BUFFER PTR OFFSET
03B66 2622                    INCR     R2,2             ;STEP TO NEXT RMODEM BUFFER (A,B,C,A,..)
```

```
03B67 1A20               SKPL    LEZ
03B68 9021FFFC           MOVIR   R2,-4
03B6A E0200546           MOVRM   R2,RMPO
03B6C E200053F           CMPMZ   RUN
03B6E 80103BC2           JMP     RMI$RTN,EQZ       ;DO NIL IF VOCODER NOT RUNNING
03B70 90943B60           MOVIR   R1,@RMPTR(R2)     ;R1 NOW POINTS TO LATEST RMODEM BUFFER
03B72 70320000           MOVNE   R3,R1,IDW$ONHK    ;GET ON-HOOK BIT FROM 1ST RMODEM DATUM
03B74 E0300551           MOVRM   R3,RONHK          ; AND SAVE IT FOR USE BY ADANINT ROUTINE
03B76 F040057F           MOVMR   R4,RERRS          ;RERRS=# OF CHANNEL ERRORS PER BUFFER
03B78 1A10               SKPL    EQZ               ; TO BE SIMULATED
03B79 86103BCA           CALL    R1,RMI$SIM
03B7B E2000552           CMPMZ   RSYNC             ;WHAT DO WE KNOW ABOUT SYNC?
03B7D 0800               EVEN
03B7E 80103BB6           JMP     RMI$SINIT,EQZ     ;0 MEANS NIL, SO INIT
03B80 80303BC0           JMP     RMI$SRCH,LTZ      ;-1 MEANS WE LACK IT, SO SEARCH
03B82 86103C52           CALL    R1,SYNCUPDATE     ;+1 MEANS WE HAVE IT, SO UPDATE
03B84 E2000552           CMPMZ   RSYNC             ;DO WE STILL HAVE IT AFTER THE UPDATE?
03B86 80103BBA           JMP     RMI$LOST,EQZ      ;NO, SO REINIT FOR SYNC-SEARCHING
03B88 F0300547           MOVMR   R3,RBTPO          ;R3 GETS RBITS POINTER OFFSET
03B8A E2863B58           CMPMZ   @RBTFPTR(R3)      ;IS RBITS BUFFER AVAILABLE (EMPTY)?
03B8C 81103BC6           JMP     RMI$DISC,NEZ      ;NO, SO DISCARD THE FRAME OF DATA
                         ;COPY FRAME OF BITS (INCLUDING SYNC BIT) FROM PREVIOUS AND CURRENT
                         ;  RMODEM BUFFERS TO RBITS BUFFER.  WE COPY (MBLNGTH-RBOFO) BITS
                         ;  FROM (PREV. RMODEM + RBOFO) TO RBITS, AND THEN (RBOFO) BITS FROM
                         ;  (CURRENT RMODEM) TO (RBITS + MBLNGTH-RBOFO).
                         ;NOTE THAT WE JAM ADDREASES INTO TWO BMOVE INSTRUCTIONS BELOW!
03B8E 90943B5E           MOVIR   R1,@RMPTR-2(R2)   ;R1 = PTR TO PREVIOUS RMODEM BUFFER
03B90 FC100553           ADDMR   R1,RBOFO          ;R1 NOW POINTS TO THE SYNC BIT
03B92 2711               DECR    R1,1              ;MAKE R1 BE PTR-1 FOR BMOVE
03B93 0800               EVEN
03B94 90C63E0C           MOVIR   R4,@RBTBPTR(R3)   ;R4 GETS PTR TO RBITS BUFFER
03B96 EF403BA0           SAF     R4,RMI$BMO1       ;SET UP 1ST BMOVE WITH 17-BIT ADR
03B98 90500209           MOVIP   R5,MBLNGTH-1      ;R5 GETS NUMBER-1 OF BITS TO BE MOVED
03B9A FF500553           SUBMR   R5,RBOFO          ; FROM PREVIOUS RMODEM BUFFER
03B9C 9C4A0001           ADDIR   R4,1(R5)          ;R4 NOW POINTS TO RBITS FOR 2ND COPY
03B9E EF403BAA           SAF     R4,RMI$BMO2       ;SET UP 2ND BMOVE WITH 17-BIT ADR
03BA0 D51A0000 RMI$BMO1: BMOVE   R1,R5,0           ;(ADR GETS SET ABOVE)
03BA2 F0500553           MOVMR   R5,RBOFO          ;DO WE NEED ANY BITS FROM 2ND BUFFER?
03BA4 80103BAC           JMP     RMI$STF,EQZ       ;JUMP IF NO
03BA6 90943B60           MOVIR   R1,@RMPTR(R2)     ;R1=PTR TO CURRENT RMODEM BUFFER
03BA8 2711               DECR    R1,1              ;PTR-1 FOR BMOVE
03BA9 2751               DECR    R5,1              ;NUMBER-1 FOR BMOVE
03BAA D51A0000 RMI$BMO2: BMOVE   R1,R5,0           ;(ADR GETS SET ABOVE)
03BAC E5863B58 RMI$STF:  INCM    @RBTFPTR(R3)      ;SET RBTFLAG TO MARK RBITS BUFFER FULL
03BAE 9431FFFE           XORIR   R3,-2             ;SWITCH RBITS BUFFERS (-2,0,-2,...)
03BB0 E0300547           MOVRM   R3,RBTPO
03BB2 E500054F           INCM    RFCTR             ;COUNT FRAME RECEIVED
03BB4 2000               HOP     RMI$RTN
                         ;
03BB5 0800               EVEN
03BB6 86103BEA RMI$SINIT: CALL   R1,SYNCINIT       ;INIT THE SYNC-SEARCH
03BB8 2009               HOP     RMI$RTN
                         ;
```

```
03BB9 0000                  EVEN
03BBA 9A943B5E RMI$LOST: MOVIR R1,@RMPTR-2(R2)  ;WE HAVE LOST SYNC:
03BBC 861B3BE1           CALL    R1,SYNCINIT       ; CALL SYNCINIT ON THE OLDER RMODEM
03BBE 90943B60           MOVIR   R1,@RMPTR(R2)     ; BUFFER, THEN SYNCSRCH ON THE NEWER ONE
03BC0 861B3C12 RMI$SRCH: CALL  R1,SYNCSRCH
03BC2 F80DFFCE RMI$RTN: MOVLM  CLR+G2,SYS$FLGS  ;MARK END OF RMODENINT
03BC4 0E70               RETURN
                ;
                ;THE RBITS BUFFER ISN'T READY (EMPTY) YET, SO WE HAVE NO PLACE TO
                ;  PUT THE RMODEM DATA, AND WE HAVE TO DISCARD IT.
03BC5 0000               EVEN
03BC6 E500054C RMI$DISC: INCM  RMFDC           ;COUNT RMODEM FRAME DISCARD
03BC8 2107               HOP     RMI$RTN
                ;
                ;ROUTINE TO SIMULATE CHANNEL ERRORS.  WE USE THE EXEC CODE AS A "RANDOM"
                ;  NUMBER GENERATOR.  WE MAINTAIN A POINTER TO THAT PART OF BUS 1.  FOR
                ;  EACH ERROR, WE PULL IN 2 WORDS, XOR, AND OFF TO 9 BITS, +10, AND USE
                ;  THAT AS AN OFFSET INTO THE CURRENT RMODEM BUFFER, WHERE WE FLIP THE
                ;  DATA BIT.  THUS WE CAN ERROR ANY OF THE LSST 512 BITS OF THE FRAME.
                ;  DOING IT FIVE TIMES PER FRAME GIVES 0.96% ERRORS.
                ;UPON ENTRY, R1 POINTS TO THE START OF THE RMODEM BUFFER, AND R4 HOLDS
                ;  THE NUMBER OF ERRORS TO BE MADE.
03BC9 0000               EVEN
03BCA 2741     RMI$SIM:  DECR    R4,1              ;NUMBER-1 OF ERRORS
03BCB 0000               EVEN
03BCC F05F3BE6           MOVMR   R5,ERRPTR         ;PICK UP POINTER INTO EXEC CODE
03BCE 90700001           MOVIR   R7,IDW$RD         ;WILL BE USED TO FLIP DATA BIT
03BD0 F06A0000 01:       MOVMR   R6,0(R5)          ;GET "RANDOM" NUMBER
03BD2 F46A0001           XORMR   R6,1(R5)          ;MAKE IT MORE SO
03BD4 2663               INCR    R6,3              ;AVOID BIAS FOR 0
03BD5 2653               INCR    R5,3              ;STEP RANDOM NUMBER PTR
03BD6 9A6001FF           ANDIR   R6,$1FF           ;MAKE THE RANDOM NUMBER 9 BITS
03BD8 266A               INCR    R6,10             ;GIVE IT THE RANGE 10-521
03BD9 4C62               ADDRR   R6,R1             ;MAKE PTR INTO RMODEM BUFFER
03BDA F57C0000           XORRM   R7,0(R6)          ;FLIP DATA BIT FOR THE ERROR
03BDC 8C403BD0           DJP     R4,01             ;GO BACK IF NOT ENOUGH ERRORS YET
03BDE 92501C77           CMPIR   R5,RAND$U         ;CHECK IF ERRPTR NEEDS RESETTING
03BE0 1B20               SKPL    LE
03BE1 905000E8           MOVIR   R5,RAND$L         ;RESET ERRPTR TO LOWER VALUE
03BE3 E05F3BE6           MOVRM   R5,ERRPTR
03BE5 0E70               RETURN
03BE6 00E8     ERRPTR:   DATA    RAND$L
```

```
                      * FRAME SYNCHRONIZATION ROUTINES
                      ;
                      ; MISC. INTERNAL VARIABLES
03BF7 0000            ERRSYNC: DATA    0               ;COUNTS SYNC ERRORS
03BF8 0000            LASTERR: DATA    0               ;=1 IF SYNC ERROR ON PREVIOUS FRAME
03BF9 0000            OLDSYNC: DATA    0               ;LSB = SYNC BIT ON PREVIOUS FRAME
                      ;RBOFO: (1ST 8I)                 ;OFFSET FROM START OF RMODEM BUFFER TO
                      ;                                ; POSITION WHERE SYNC BIT WAS FOUND
                      ;
                      ;
                      ; SYNCINIT: INITIALIZE BUFFERS, ETC. FOR SYNC SEARCHING.  CALL WITH
                      ;    R1 POINTING TO RMODEM BUFFER.
                      ; ALSO ZEROS THE GAIN WORD OF BOTH RSOURCE BUFFERS AND ZEROS THE
                      ;    ENTIRE "LAST" RBITS BUFFER, SINCE IF WE JUST LOST SYNC, THOSE
                      ;    BUFFERS CONTAIN GARBAGE, AND WE DON'T WANT TO RESUME SYNTHESIZING
                      ;    WITH THEM.
                                 EVEN
03BFA 90200209        SYNCINIT: MOVIR  R2,NBLNGTH-1    ;BUFFER OFFSET = LAST WORD IN BUFFER
03BFC 4C14                      ADDRR  R1,R2           ;R1 NOW POINTS TO END OF RMODEM BUFFER
03BED 70320001        SI$LOOP:  MOVNK  R3,R1,1         ;PICK UP DATA BIT (BIT 0)
03BEF E0347BEC                  MOVRN  R3,RSSPF(R2)    ; AND STASH IT IN RSSPF(I)
03BF1 2711                      DECR   R1,1
03BF2 8C203BED                  DJP    R2,SI$LOOP
03BF4 E0200552                  MOVRM  R2,RSYNC        ;SET RSYNC=-1 TO MEAN SEARCH FOR SYNC
03BF6 CF000960                  MOVZML RSRA+NRESID*W$  ;ZERO THE 217TH FULLWORD (=GAIN) OF BOTH
03BF8 CE00002C                  MOVZML RSRB+NRESID*W$  ; RSOURCE BUFFERS
03BFA F0300547                  MOVNR  R3,RBTPO        ;PICK UP PTR OFFSET TO "THIS" RBITS
03BFC 9431FFFE                  XORIR  R3,-2           ;SWITCH IT TO THE "OTHER" ONE
03BFE 90C63E0C                  MOVIR  R4,@RBTBPTR(R3) ;R4 NOW POINTS TO "OTHER" RBITS BUFFER
03C00 CE000000                  MOVZML 0(R4)           ;CLEAR THE 1ST FULLWORD
03C02 2642                      INCR   R4,W$           ;RBITS+2 FOR BMOVEL DEST. ADR
03C03 EF403C08                  SAF    R4,SI$BMO       ;SET UP ADR IN BMOVEL BELOW
03C05 2744                      DECR   R4,2*W$         ;RBITS-2 FOR BMOVEL SOURCE ADR
03C06 90200103                  MOVIR  R2,(NBLNGTH/2)-2 ;SET UP TO CLEAR NBLNGTH HALFWORDS
03C08 D7440000        SI$BMO:   BMOVEL R4,R2,0         ;(ADR IS SET UP ABOVE)
03C0A CE007DF6                  MOVZML RSSSS           ;CLEAR THE 1ST FULLWORD OF RSSSS
03C0C 90407DF4                  MOVIR  R4,RSSSS-W$     ;CLEAR THE RSSSS BUFFER TOO
03C0E D7447DF8                  BMOVEL R4,R2,RSSSS+W$  ;(R2 IS STILL SET FROM PREVIOUS BMOVEL)
03C10 0E70                      RETURN
                      ;
                      ;
                      ; SYNCSRCH: SEARCH INCOMING RMODEM BUFFER FOR SYNC, AND IF SYNC
                      ;    IS FOUND, SET RSYNC TO +1 AND SET RBOFO TO THE FRAME OFFSET
                      ;    CORRESPONDING TO THE LOGICAL START OF THE FRAME.
                      ; CALL WITH R1 POINTING TO THE START OF THE RMODEM BUFFER TO BE
                      ;    SEARCHED.
                      ; REGISTER USAGE:
                      ;    R1 - PTR TO THE RMODEM BUFFER BEING SEARCHED
                      ;    R2 - OFFSET INTO THE RMODEM, RSSPF, RSSSS BUFFERS
                      ;    R3 - SCRATCH AC
                      ;    R4 - MAXCNT, THE MAX. OF RSSSS(I)
                      ;    R5 - NMAX, THE # OF INSTANCES OF MAXCNT IN RSSSS(I)
                      ;    R6 - IMAX, THE FRAME OFFSET WHERE MAXCNT WAS FOUND
```

```
                    (00907) ; VARIABLES:
                    (00908) ;   RBOFO - BEGINNING OF FRAME OFFSET TO THE SYNC WORD IN THE
                    (00909) ;               RMODEM BUFFER
                    (00910) ; BUFFERS:
                    (00911) ;   RSSPF - SYNC-SEARCH PREVIOUS FRAME - STORES THE DATA BIT OF THE
                    (00912) ;               PREVIOUS RMODEM BUFFER
                    (00913) ;   RSSSS - SYNC-SEARCH SUM-OF-SYNC COUNTS - FOR EACH RMODEM WORD,
                    (00914) ;               STORES THE NUMBER OF CONSECUTIVE FRAMES OF DATA BIT
                    (00915) ;               ALTERNATION.
03C11 0000          (00916)         EVEN
03C12 90200209      (00917) SYNCSRCH: MOVIR R2,MOLNGTH-1    ;INIT BUFFER OFFSET TO LAST IN FRAME
03C14 4C14          (00918)         ADDRR   R1,R2           ;MAKE R1 POINT THERE TOO
03C15 0D40          (00919)         CLR     R4              ;MAXCNT <- 0
03C16 70320001      (00920) SS$LOOP: MOVMK  R3,R1,IDW$RD    ;R3 <- NEW DATA BIT (OTHER BITS MASKED)
03C18 F2347BEC      (00921)         CMPMR   R3,RSSPF(R2)    ;DOES THIS BIT ALTERNATE FROM LAST?
03C1A B0103C2C      (00922)         JMP     SS$NOALT,EQ     ;JUMP IF NO ALTERNATION
03C1C E5047DF6      (00923) SS$ALT: INCM    RSSSS(R2)       ;YES: COUNT THE BIT ALTERNATION
03C1E F2447DF6      (00924)         CMPMR   R4,RSSSS(R2)    ;IS MAXCNT >= RSSSS(I)?
03C20 81303C28      (00925)         JMP     NOMAX,GE        ;YES
03C22 F0447DF6      (00926) NEWMAX: MOVMR   R4,RSSSS(R2)    ;NO: RSSSS(I) BECOMES THE NEW MAXCNT
03C24 90500001      (00927)         MOVIR   R5,1            ;NMAX <- 1 (UNIQUE SO FAR)
03C26 4064          (00928)         MOVRR   R6,R2           ;AND WE SAVE THE OFFSET IN IMAX
03C27 2006          (00929)         HOP     SS$LPTST
                    (00930) ;
                    (00931)         EVEN
03C28 81103C2E      (00932) NOMAX:  JMP     SS$LPTST,NE     ;MAXCNT = RSSSS(I)?
03C2A 2651          (00933)         INCR    R5,1            ;YES, SO THE MAX ISN'T UNIQUE
03C2B 2002          (00934)         HOP     SS$LPTST
                    (00935) ;
                    (00936)         EVEN
03C2C CC047DF6      (00937) SS$NOALT: MOVZM RSSSS(R2)       ;CLEAR RSSSS(I)
03C2E E0347BEC      (00938) SS$LPTST: MOVRM R3,RSSPF(R2)    ;SAVE NEW DATA BIT IN RSSPF(I)
03C30 2711          (00939)         DECR    R1,1            ;DECR RMODEM BUFFER PTR
03C31 0000          (00940)         EVEN
03C32 BC203C16      (00941)         DJP     R2,SS$LOOP      ;DECR OFFSET AND TEST IF DONE BUFFER
                    (00942) ; WE HAVE NOW SCANNED THE WHOLE BUFFER.  IS MAXCNT >= THE SYNC
                    (00943) ;   ACQUISITION THRESHOLD, AND IT IT A UNIQUE MAX?  IF SO, WE HAVE
                    (00944) ;   FOUND SYNC.
03C34 9240000A      (00945)         CMPIR   R4,ACQTHR       ;MAXCNT >= ACQTHR?
03C36 80303C4E      (00946)         JMP     SS$NO,LT        ;NO
03C38 92500001      (00947)         CMPIR   R5,1            ;YES: IS NMAX=1?
03C3A B1103C4E      (00948)         JMP     SS$NO,NE        ;NO
03C3C E0500552      (00949)         MOVRM   R5,RSYNC        ;YES: WE HAVE SYNC. SET RSYNC TO +1
03C3E E0600553      (00950)         MOVRM   R6,RBOFO        ;SAVE OFFSET IN RBOFO
03C40 90300004      (00951)         MOVIR   R3,LOSETHR
03C42 E0303BE7      (00952)         MOVRM   R3,ERRSYNC      ;INIT ERRSYNC TO COUNT ERRORS
03C44 CC003BE0      (00953)         MOVZM   LASTERR         ; FOR SYNCUPDATE NEXT FRAME
03C46 9C1C0001      (00954)         ADDIR   R1,1(R6)        ;R1 POINTS TO WORD HOLDING SYNC
03C48 70320001      (00955)         MOVMK   R3,R1,IDW$RD    ;PICK UP SYNC BIT AND SAVE IT IN
03C4A E03F3BE9      (00956)         MOVRM   R3,OLDSYNC      ; OLDSYNC FOR USE IN SYNCUPDATE ON NEXT
03C4C 0F70          (00957)         RETURN                  ; FRAME
                    (00958) ;
03C4D 0000          (00959)         EVEN
```

```
03C4E E0200552 SS$NO:  MOVRM   R2,RSYNC        ;NO SYNC YET: SET TO -1 (=SEARCH)
03C50 0E70             RETURN                  ; FOR GOOD MEASURE
               ;
               ;
               ; SYNCUPDATE:  CHECK RMODEM BUFFER TO SEE IF THE DATA BIT AT THE
               ;   EXPECTED SYNC POSITION (START OF BUFFER+RBOFO) HAS THE EXPECTED
               ;   VALUE.  IF THERE ARE LOSETHR ERRORS WITHOUT 2 CONSECUTIVE CORRECT
               ;   COMPARISONS, CLEAR RSYNC TO SIGNIFY LOSS OF FRAME SYNCHRONIZATION.
               ; ENTER WITH R1 POINTING TO THE LATEST RMODEM BUFFER.  ALSO, THE
               ;   WORD RBOFO HOLDS THE OFFSET IN THE FRAME THAT POINTS TO THE
               ;   SYNC WORD TO BE VERIFIED.
               ; VARIABLES:
               ;   RBOFO - BEGINNING OF FRAME OFFSET
               ;   OLDSYNC - SYNC BIT FROM PREVIOUS FRAME
               ;   LASTERR - NZ IF SYNC ERROR ON PREVIOUS FRAME
               ;   ERRSYNC - COUNTS SYNC ERRORS (FROM LOSETHR DOWN TO 0)
03C51 0000             EVEN
03C52 FC100553 SYNCUPDATE: ADDMR R1,RBOFO      ;R1 NOW POINTS TO NEW SYNC WORD
03C54 6022             MOVWR   R2,R1           ;PICK UP THE NEW SYNC WORD
03C55 0000             EVEN
03C56 F4203BE9         XORMR   R2,OLDSYNC      ;XOR WITH LAST SYNC BIT
03C58 2C04             SRBS    0,R2            ;SKIP IF DATA BIT SET (GOOD SYNC)
03C59 200C             NOP     SU$ERR
03C5A E2003BE8         CMPNZ   LASTERR         ;WAS THERE AN ERROR LAST FRAME (NEZ)?
03C5C 81103C62         JMP     SU$ELF,NEZ      ;YES
03C5E 90300004         MOVIR   R3,LOSETHR      ;NO:  THAT'S (AT LEAST) 2 CONSECUTIVE
03C60 E0303BE7         MOVRM   R3,ERRSYNC      ; CORRECT FRAMES, SO RESET ERROR COUNT
03C62 CC003BE8 SU$ELF: MOVZM   LASTERR         ;REMEMBER NONERROR FOR NEXT FRAME
03C64 2009             NOP     SU$VALID
               ;
03C65 0000             EVEN
03C66 90200001 SU$ERR: MOVIR   R2,1            ;COME HERE ON SYNC BIT ERROR
03C68 E0203BE8         MOVRM   R2,LASTERR      ;REMEMBER ERROR FOR NEXT FRAME
03C6A E4003BE7         DECM    ERRSYNC         ;HAVE WE COUNTED LOSETHR ERRORS?
03C6C 81203C74         JMP     SU$LOSE,LEZ     ;YES, THEREFORE WE'VE LOST SYNC
03C6E 90100001 SU$VALID: MOVIR R1,IDW$RD       ;COMPLEMENT OLDSYNC IN PREPARATION
03C70 F5103BE9         XORRM   R1,OLDSYNC      ; FOR USE NEXT FRAME
03C72 0E70             RETURN
               ;
03C73 0000             EVEN
03C74 CC000552 SU$LOSE: MOVZM  RSYNC           ;ZERO RSYNC TO SAY WE LOST SYNC
03C76 E5000550         INCM    RLCTR           ;COUNT LOSS OF SYNC
03C78 0E70             RETURN
```

```
                     'AOMINT: AOM INTERRUPT SERVICE ROUTINE
                     ; JJW, 3 OCT 79
                     ;
                     ;AOMINT COPIES SYNTHESIS WAVEFORM DATA FROM AN RSINK BUFFER TO A D/A
                     ;  BUFFER.  IF THE RSINK BUFFER ISN'T READY YET, A FRAME OF SILENCE
                     ;  IS COPIED FROM RSNKC INSTEAD.
                     ;POINTER OFFSETS:
                     ;  RSNPO IS FOR RSINK BUFFERS AND FLAGS (-2,0; INIT TO 0 SO IT POINTS
                     ;    TO RSNKB THE FIRST TIME).  USE RSIBPTR(RSNPO) TO POINT TO THE
                     ;    RSINK BUFFER AND RSIFPTR(RSNPO) FOR THE FLAG.
                     ;  DAPO IS FOR D/A BUFFERS (-2,0; INIT TO 0 SO IT GETS SWITCHED TO
                     ;    DA$CPYA THE FIRST TIME).  USE DABPTR(DAPO) TO POINT TO THE
                     ;    D/A BUFFER COPYING ROUTINE.
                     ;
03C79 0000                     EVEN
03C7A 00003CAE                 ADDR    DA$CPYA          ;ROUTINE TO COPY TO D/A BUFFER A
03C7C 00003CB0       DABPTR:   ADDR    DA$CPYB          ;ROUTINE TO COPY TO D/A BUFFER B
                     ;
03C7E 00000B46                 ADDR    RSNKA-2          ;RSINK A BUFFER (-2 FOR THE BMOVEL)
03C80 00000C18       RSIBPTR:  ADDR    RSNKB-2          ;RSINK B BUFFER
03C82 0000053D                 ADDR    RSNFA            ;RSINK A FLAG (0=EMPTY, 1=FULL)
03C84 0000053E       RSIFPTR:  ADDR    RSNFB            ;RSINK B FLAG
                     ;
                               EVEN
03C86 F040FFCE       AOMINT:   MOVLM   SET+G2,SYS$FLGS  ;MARK START OF AOMINT
03C88 F010549        MOVMR     R1,DAPO                  ;R1 GETS D/A BUFFER PTR OFFSET
03C8A 9411FFFE                 XORIR   R1,-2            ;SWITCH TO OTHER BUFFER (0,-2,0,...)
03C8C E0100549                 MOVRM   R1,DAPO
03C8E E200053F                 CMPMZ   RUN
03C90 80103CA2                 JMP     AOM$RTN,EQZ      ;DO NIL IF VOCODER NOT RUNNING
03C92 F030054B                 MOVMR   R3,RSNPO         ;R3 GETS RSINK PTR OFFSET
03C94 90A63C80                 MOVIR   R2,@RSIBPTR(R3)  ;R2 POINTS TO RSINK BUFFER-1
03C96 E2863C84                 CMPMZ   @RSIFPTR(R3)     ;IS THE RSINK BUFFER READY (FULL)?
03C98 80103CA6                 JMP     AOM$NRDY,EQZ     ;JUMP IF NOT READY
03C9A 06823C7C                 CALL    R0,@DABPTR(R1)   ;COPY FROM RSINK TO D/A BUFFER
03C9C CC863C84                 MOVZM   @RSIFPTR(R3)     ;CLR RSINK FLAG TO SHOW BUFFER EMPTY
03C9E 9431FFFE                 XORIR   R3,-2            ;SWITCH RSINK BUFFERS (-2,0,-2,...)
03CA0 E030054B                 MOVRM   R3,RSNPO
03CA2 F000FFCE       AOM$RTN:  MOVLM   CLR+G2,SYS$FLGS  ;MARK END OF AOMINT
03CA4 0E70                     RETURN
                     ;
03CA5 0000                     EVEN
03CA6 E500054D       AOM$NRDY: INCM    RSNNR            ;COUNT RSINK BUFFER NOT READY
03CA8 9020BCF6                 MOVIR   R2,RSNKC-2       ;COPY SILENCE FROM RSNKC TO D/A BUFFER
03CAA 06823C7C                 CALL    R0,@DABPTR(R1)
03CAC 2100                     HOP     AOM$RTN          ;RETURN WITHOUT SWITCHING RSINK BUFFS
                     ;
03CAD 0000                     EVEN
03CAE 90400023       DA$CPYA:  MOVIR   R4,SL6-1
03CB0 D72B0D00                 BMOVEL  R2,R4,RDABA
03CB2 D720BE10                 BMOVEL  R2,R4,RDABA+2*SL6
03CB4 D720BF60                 BMOVEL  R2,R4,RDABA+4*SL6
03CB6 0E70                     RETURN
```

```
                   (01057) ;
03CB7 0000         (01058)               EVEN
03CB8 904F0023     (01059) DASCPYB: MOVIB   R4,SL6-1
03CBA D720BEA0     (01060)               BMOVEL  R2,R4,RDABB
03CBC D720BEF0     (01061)               BMOVEL  R2,R4,RDABB+2*SL6
03CBE D720BF30     (01062)               BMOVEL  R2,R4,RDABB+4*SL6
03CC0 0E70         (01063)               RETURN
                   (01064)
```

```
"PRORES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE XNTR FRAME.
;
;PROTECT TAKES QUANTIZED AND CODED ANALYSIS PARAMETERS AND RESIDUAL
;  SAMPLES FROM A TSINK BUFFER, AND FORMS A BITSTREAM, INCLUDING
;  ERROR-PROTECTING CODES FOR SOME BITS OF CERTAIN PARAMETERS, IN
;  A TBITS BUFFER.
;THE "PROTECT" MODULE HAS BEEN BROKEN INTO 2 SHORTER ONES, WHICH
;  CAN BE (MOSTLY) HIDDEN UNDER APU MODULE EXECUTIONS.
;    PRORES(AB) -- BITSTREAMS THE RESIDUAL SAMPLES (ABOUT 3.7 MS)
;    PROPAR(AB) -- ERROR-PROTECTS AND BITSTREAMS THE OTHER PARAMETERS
;                  (ABOUT 0.6 MS).  PROPAR ALSO ACCUMULATES HISTOGRAM
;                  DATA FOR THE G, DG, P, C1-C3, K1-K6 PARAMETERS.
;
;THESE ROUTINES ARE ENTERED WITH R1 POINTING TO THE FCB0.  1(R1)
;  IS THE BUFFER/FLAG/ROUTINE POINTER OFFSET, -2 FOR "A" OR 0 FOR "B".
;
;FORMAT OF INPUT (TSINK) BUFFER IS (ONE DATUM PER HALF-WORD):
;  216 RESIDUAL SAMPLES (0/2)
;  GAIN (6/6)
;  DG1 - DG3 (2/2)
;  PITCH (7/7)
;  C1 (2/3)
;  C2 (3/4)
;  C3 (2/3)
;  K1 (5/6)
;  K2 (4/5)
;  K3 (3/4)
;  K4-K6 (2/4)
;WHERE (N/M) MEANS N HIGH-ORDER BITS PROTECTED OUT OF M TOTAL FOR PARAM.
;
;
;FORMAT OF OUTPUT (TBITS) BUFFER IS (ONE BIT PER HALF-WORD):
;  POSITION FOR THE SYNC BIT (IN THE TMODEM BUFFER)
;  RES1 - RES216 (2 BITS EACH, LSB FIRST)
;  B5432 GAIN (7 BITS, HAMMING (7,4) CODE)
;  B10 GAIN, B10 DG1 (7)
;  B10 DG2, B10 DG3 (7)
;  B6543 PITCH (7)
;  B0 C1 (1)
;  B2 C1, B210 PITCH (7)
;  B0 C2 (1)
;  B321 C2, B1 C1 (7)
;  B0 C3 (1)
;  B0 K1 (1)
;  B54 K1, B21 C3 (7)
;  B0 K2 (1)
;  B4321 K2 (7)
;  B0 K3 (1)
;  B3 K3, B321 K1 (7)
;  B32 K4, B21 K3 (7)
;  B10 K4 (2)
;  B10 K5 (2)
;  B10 K6 (2)
```

```
PAGE   32: [BGND]<DCA16>BMN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF
       PRORES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE XMTR FRAME.

                  ;  B32 K6, B32 K5 (7)
                  ;
                  ;REGISTER USAGE:
                  ;  R1 = PTR-1 TO INPUT (TSINK) BUFFER
                  ;  R2 = PTR TO OUTPUT (TBITS) BUFFER
                  ;  R3 = COUNTER IN PRBRLP
                  ;  R4-R6 ARE USED FOR HOLDING PIECES OF PARAMETERS.  R6 IS
                  ;     ALSO USED IN OUT7, BUT THE RETURN RESTORES IT.
                  ;  R7 = HOLDS DATUM FOR INPUT TO OUT7, OUT2 ROUTINES.
                  ;
                  ;POINTERS TO BUFFERS AND FLAGS.  SOME OF THESE ARE ALSO USED BY
                  ;  TMODEMINT.
03CC1 0000                  EVEN
03CC2 0000A17F              ADDR    TSNKA-1            ;TSINK BUFFERS (-1 FOR POPXI'S)
03CC4 0000A265    TSNBPTR:  ADDR    TSNKB-1
                  ;
03CC6 0000A34D              ADDR    TBTA+1             ;TBITS BUFFERS (BITSTREAM STARTS AT +1
03CC8 0000A557    TBTBPTR:  ADDR    TBTB+1             ; TO ALLOW FOR SYNC BIT IN TMDM BUFFS)
                  ;
                  ;
03CCA F840FFCE    PRORES:   MOVLM   SET+G1,SYS$FLGS    ;FOR SCOPE DISPLAY ...
03CCC F0320001              MOVMR   R3,1(R1)           ;PICK UP POINTER OFFSET
03CCE 90963CC4              MOVIR   R1,@TSNBPTR(R3)    ;R1 GETS PTR-1 TO TSINK BUFFER
03CD0 90A63CC8              MOVIR   R2,@TBTBPTR(R3)    ;R2 GETS OUTPUT PTR (TO TBT + 1)
03CD2 903000D7              MOVIR   R3,NRESID-1        ;R3=RESIDUAL SAMPLE COUNTER
03CD4 301F        01:       POPXI   R1,R7              ;PICK UP 2-BIT CODED RESIDUAL SAMPLE
03CD5 0000                  EVEN
03CD6 506E0001              MOVKR   R6,R7,1            ;EXTRACT LSB
03CD8 266C                  INCR    R6,TMBITS          ;ADD THE TMBITS
03CD9 342C                  PUSHXI  R2,R6              ;AND OUTPUT IT
03CDA 3C71                  LRS     R7,1               ;MOVE THE MSB INTO POSITION
03CDB 267C                  INCR    R7,TMBITS          ;ADD THE TMBITS
03CDC 342E                  PUSHXI  R2,R7              ;AND OUTPUT THE MSB
03CDD 0000                  EVEN
03CDE 8C3A3CD4              DJP     R3,01              ;LOOP IF MORE RESIDUALS TO DO
                  ;
03CE0 F800FFCE              MOVLM   CLR+G1,SYS$FLGS
03CE2 0E70                  RETURN
                  ;
                  ;
03CE3 0000                  EVEN
03CE4 F840FFCE    PROPAR:   MOVLM   SET+G1,SYS$FLGS    ;FOR SCOPE DISPLAY ...
03CE6 F0320001              MOVMR   R3,1(R1)           ;PICK UP POINTER OFFSET
03CE8 90963CC4              MOVIR   R1,@TSNBPTR(R3)    ;R1 GETS PTR-1 TO TSINK BUFFER
03CEA 9C1000D8              ADDIR   R1,NRESID*H$       ;STEP OVER NRESID CODED RESIDUAL SAMPLES
03CEC 90A63CC8              MOVIR   R2,@TBTBPTR(R3)    ;R2 GETS OUTPUT PTR (TO TBT + 1)
03CEE 9C2001B0              ADDIR   R2,NRESID*2*H$     ;STEP OVER SPACE FOR NRESID*2 RESIDUAL BITS
                  ;
03CF0 301F                  POPXI   R1,R7              ;GAIN (6 MSB'S PROTECTED/6 TOTAL BITS)
03CF1 0000                  EVEN
03CF2 E50F9C00              INCM    GHIST(R7)          ;INCREMENT BIN FOR GAIN HISTOGRAM
03CF4 505F0003              MOVKR   R5,R7,R10          ;SAVE LOW 2 BITS IN R5
03CF6 3C72                  LRS     R7,2               ;SHIFT 4 HIGH BITS INTO POSITION
```

```
03CF7 0000     (01171)          EVEN
03CF8 8660309C (01172)          CALL    R6,OUT7          ;AND OUTPUT THEM IN HAMMING(7,4) CODE
               (01173) ;
03CFA 301E     (01174)          POPXI   R1,R7            ;DELTA-GAIN-1 (2/2)
03CFB 0000     (01175)          EVEN
03CFC E50F9C48 (01176)          INCM    DGHIST(R7)
03CFE 3A52     (01177)          LLS     R5,2             ;SHIFT LOW 2 BITS OF GAIN
03CFF 467A     (01178)          IORRR   R7,R5            ;COMBINE WITH THE 2 BITS OF DG1
03D00 8660309C (01179)          CALL    R6,OUT7          ;OUTPUT PROTECTED B10 GAIN, B10 DG1
               (01180) ;
03D02 301E     (01181)          POPXI   R1,R7            ;DG2 (2/2)
03D03 0000     (01182)          EVEN
03D04 E50F9C48 (01183)          INCM    DGHIST(R7)
03D06 3A72     (01184)          LLS     R7,2             ;SHIFT OVER TO MAKE ROOM FOR DG3
               (01185) ;
03D07 301C     (01186)          POPXI   R1,R6            ;DG3 (2/2)
               (01187)          EVEN
03D08 E50C9C48 (01188)          INCM    DGHIST(R6)
03D0A 467C     (01189)          IORRR   R7,R6            ;COMBINE DG2, DG3 BITS
03D0B 0000     (01190)          EVEN
03D0C 8660309C (01191)          CALL    R6,OUT7          ;OUTPUT PROTECTED B10 DG2, B10 DG3
               (01192) ;
03D0E 301E     (01193)          POPXI   R1,R7            ;PITCH (7/7)
03D0F 0000     (01194)          EVEN                     ;(WE DON'T DO HISTOGRAM FOR PITCH)
03D10 505E0007 (01195)          MOVKR   R5,R7,B210       ;SAVE 3 LSBS OF PITCH IN R5
03D12 3C73     (01196)          LRS     R7,3             ;SHIFT 4 MSBS INTO POSITION
03D13 0000     (01197)          EVEN
03D14 8660309C (01198)          CALL    R6,OUT7          ;OUTPUT PROTECTED 4 MSBS OF PITCH
               (01199) ;
03D16 301C     (01200)          POPXI   R1,R6            ;C1 (2/3)
03D17 0000     (01201)          EVEN
03D18 E50C9C4C (01202)          INCM    C1HIST(R6)
03D1A 507C0001 (01203)          MOVKR   R7,R6,B0         ;LOWEST BIT
03D1C 9670000C (01204)          IORIR   R7,TMBITS        ;ADD THE MODEM BITS
03D1E 342F     (01205)          PUSHXI  R2,R7            ;AND OUTPUT B0 OF C1 BY ITSELF
03D1F 4C6C     (01206)          ADDRR   R6,R6            ;LEFT SHIFT 1
03D20 565C0008 (01207)          IORKR   R5,R6,B3         ;PUT B2 OF C1 ONTO 3 LSBS OF PITCH
03D22 407A     (01208)          MOVRR   R7,R5
03D23 0000     (01209)          EVEN
03D24 8660309C (01210)          CALL    R6,OUT7          ;OUTPUT PROTECTED B2 C1, B210 PITCH
               (01211) ;
03D26 3C62     (01212)          LRS     R6,2             ;MOVE B1 OF C1 INTO B0 SLOT
03D27 301A     (01213)          POPXI   R1,R5            ;C2 (3/4)
               (01214)          EVEN
03D28 E50A9C54 (01215)          INCM    C2HIST(R5)
03D2A 507A0001 (01216)          MOVKR   R7,R5,B0         ;EXTRACT B0 OF C2
03D2C 9670000C (01217)          IORIR   R7,TMBITS
03D2E 342F     (01218)          PUSHXI  R2,R7            ;AND OUTPUT IT
03D2F 0000     (01219)          EVEN
03D30 507C0001 (01220)          MOVKR   R7,R6,B0         ;GET THAT SAVED B1 OF C1
03D32 567A000E (01221)          IORKR   R7,R5,B321       ;ADD IN THE 3 MSB'S OF C2
03D34 8660309C (01222)          CALL    R6,OUT7          ;OUTPUT PROTECTED B321 OF C2, B1 C1
               (01223) ;
```

```
03D36 301C      (01224)     POPXI   R1,R6          ;C3 (2/3)
03D37 0000      (01225)     EVEN
03D38 E50C9C64  (01226)     INCM    C3HIST(R6)
03D3A 507C0001  (01227)     MOVKR   R7,R6,B0       ;EXTRACT B0
03D3C 9670000C  (01228)     IORIR   R7,TMBITS
03D3E 342E      (01229)     PUSHXI  R2,R7          ;OUTPUT B0 OF C3
03D3F 3C61      (01230)     LRS     R6,1           ;SHIFT B21 TO B10 POSITION
                (01231) ;
03D40 301A      (01232)     POPXI   R1,R5          ;K1 (5/6)
03D41 0000      (01233)     EVEN
03D42 E50A9C6C  (01234)     INCM    K1HIST(R5)
03D44 507A0001  (01235)     MOVKR   R7,R5,B0       ;EXTRACT B0 AND OUTPUT IT
03D46 9670000C  (01236)     IORIR   R7,TMBITS
03D48 342E      (01237)     PUSHXI  R2,R7
03D49 0000      (01238)     EVEN
03D4A 507A0030  (01239)     MOVKR   R7,R5,B54      ;EXTRACT B54 OF K1
03D4C 3C72      (01240)     LRS     R7,2           ;SHIFT IT INTO B32 POSITION
03D4D 0000      (01241)     EVEN
03D4E 567C0003  (01242)     IORKR   R7,R6,B10      ;ADD IN THE SAVED B21 OF C3
03D50 86603D9C  (01243)     CALL    R6,OUT7        ; AND OUTPUT THEM PROTECTED
                (01244) ;                          ;(R5 STILL HOLDS B321 OF K1)
03D52 301C      (01245)     POPXI   R1,R6          ;K2 (4/5)
03D53 0000      (01246)     EVEN
03D54 E50C9CAC  (01247)     INCM    K2HIST(R6)
03D56 507C0001  (01248)     MOVKR   R7,R6,B0       ;EXTRACT B0 AND OUTPUT IT
03D58 9670000C  (01249)     IORIR   R7,TMBITS
03D5A 342E      (01250)     PUSHXI  R2,R7
03D5B 3C61      (01251)     LRS     R6,1           ;SHIFT B4321 INTO POSITION
03D5C 407C      (01252)     MOVRR   R7,R6          ; AND OUTPUT PROTECTED
03D5D 0000      (01253)     EVEN
03D5E 86603D9C  (01254)     CALL    R6,OUT7
                (01255) ;
03D60 301C      (01256)     POPXI   R1,R6          ;K3 (3/4)
03D61 0000      (01257)     EVEN
03D62 E50C9CCC  (01258)     INCM    K3HIST(R6)
03D64 507C0001  (01259)     MOVKR   R7,R6,B0       ;EXTRACT LSB AND OUTPUT IT
03D66 9670000C  (01260)     IORIR   R7,TMBITS
03D68 342F      (01261)     PUSHXI  R2,R7
03D69 0000      (01262)     EVEN
03D6A 507A000E  (01263)     MOVKR   R7,R5,B321     ;GET B321 OF K1
03D6C 3C71      (01264)     LRS     R7,1           ;SHIFT INTO POSITION
03D6D 0000      (01265)     EVEN
03D6E 567C0008  (01266)     IORKR   R7,R6,B3       ;ADD IN THE B3 OF K3
03D70 86603D9C  (01267)     CALL    R6,OUT7        ;OUTPUT B3 K3, B321 K1 PROTECTED
03D72 3C61      (01268)     LRS     R6,1           ;SHIFT B21 OF K3 INTO B10 POSITION
                (01269) ;
03D73 301A      (01270)     POPXI   R1,R5          ;FETCH K4 (2/4)
                (01271)     EVEN
03D74 E50A9CDC  (01272)     INCM    K4HIST(R5)
03D76 507A000C  (01273)     MOVKR   R7,R5,B32      ;EXTPACT B32 OF K4
03D78 567C0003  (01274)     IORKR   R7,R6,B10      ;ADD THE B21 OF K3
03D7A 86603D9C  (01275)     CALL    R6,OUT7        ;OUTPUT PROTECTED B32 K4, B21 K3
03D7C 507A0003  (01276)     MOVKR   R7,R5,B10
```

```
03D7E 86603DA0 (01277)        CALL    R6,OUT2           ;OUTPUT B10 OF K4, UNPROTECTED
               (01278) ;
03D80 301C     (01279)        POPXI   R1,R6             ;K5 (2/4)
03D81 0000     (01280)        EVEN
03D82 E50C9CEC (01281)        INCM    K5HIST(R6)
03D84 507C0003 (01282)        MOVKR   R7,R6,B10
03D86 86603DA0 (01283)        CALL    R6,OUT2           ;OUTPUT B10 OF K5, UNPROTECTED
               (01284) ;
03D88 301A     (01285)        POPXI   R1,R5             ;K6 (2/4)
03D89 0000     (01286)        EVEN
03D8A E50A9CFC (01287)        INCM    K6HIST(R5)
03D8C 507A0003 (01288)        MOVKR   R7,R5,B10
03D8E 86603DA0 (01289)        CALL    R6,OUT2           ;OUTPUT B10 K6, UNPROTECTED
03D90 507C000C (01290)        MOVKR   R7,R6,B32         ;EXTRACT B32 OF K5
03D92 3C72     (01291)        LRS     R7,2              ;SHIFT INTO B10 POSITION
03D93 0000     (01292)        EVEN
03D94 567A000C (01293)        IORKR   R7,R5,B32         ;ADD THE B32 OF K6
03D96 86603D9C (01294)        CALL    R6,OUT7           ;OUTPUT PROTECTED B32 K6, B32 K5
03D98 F60RFFCE (01295)        MOVLM   CLR+G1,SYS$FLGS
03D9A 0E70     (01296)        RETURN
               (01297) ;
               (01298) ;OUT7 TAKES A 4-BIT DATUM IN R7 AND OUTPUTS A 7-BIT HAMMING CODE
               (01299) ;NOTE THAT WE MODIFY THE INSTRUCTION AT BMO7 BY JAMMING THE
               (01300) ;  17-BIT OUTPUT POINTER INTO ITS ADDRESS FIELD.
03D9B 0000     (01301)        EVEN
03D9C EF203DA4 (01302) OUT7:  SAF     R2,BMO7           ;STORE OUTPUT PTR IN BMOVE INSTR.
03D9E 4C7E     (01303)        ADDRR   R7,R7             ;CONVERT R7 TO FULLWORD OFFSET
03D9F 2627     (01304)        INCR    R2,7              ;UPDATE OUTPUT PTR IN R2
03DA0 90FE3DB2 (01305)        MOVIR   R7,@HPTR(R7)      ;R7 GETS PTR TO START-1
03DA2 90600006 (01306)        MOVIR   R6,7-1            ;WILL TRANSFER 7 WORDS
03DA4 D57C0000 (01307) BMO7:  BMOVE   R7,R6,0           ;MOVE 7 WORDS FROM (R7)+1 TO OUTPUT
03DA6 0E70     (01308)        RETURN
               (01309) ;
               (01310) ;OUT2 OUTPUTS 2 BITS GIVEN IN R7
03DA7 0000     (01311)        EVEN
03DA8 4C7E     (01312) OUT2:  ADDRR   R7,R7             ;CONVERT R7 TO FULLWORD OFFSET
03DA9 2622     (01313)        INCR    R2,2              ;UPDATE OUTPUT POINTER
03DAA 90FF3DD2 (01314)        MOVIR   R7,@PTR2(R7)      ;R7 GETS PTR-1
03DAC C075FFFE (01315)        POPMI   R7,-2(R2)
03DAE C075FFFF (01316)        POPMI   R7,-1(R2)
03DB0 0E70     (01317)        RETURN
               (01318)
               (01319) ;TABLE OF HAMMING CODE SEQUENCES.  INDEX WITH 2*(4-BIT VALUE)
               (01320) ;  TO BE PROTECTED; TABLE ENTRY IS PTR-1 TO THE 7-BIT HAMMING
               (01321) ;  CODEWORD.
03DB1 0000     (01322)        EVEN
03DB2 00003E01 (01323) HPTR:  ADDR H000-1
03DB4 00003DD9 (01324)        ADDR H151-1
03DB6 00003DF3 (01325)        ADDR H052-1
03DB8 00003DF8 (01326)        ADDR H103-1
03DBA 00003DE2 (01327)        ADDR H114-1
03DBC 00003DDB (01328)        ADDR H045-1
03DBE 00003DE1 (01329)        ADDR H146-1
```

```
03DC0 00003DFA (01330)          ADDR N017-1
03DC2 00003DFE (01331)          ADDR N160-1
03DC4 00003DE3 (01332)          ADDR N031-1
03DC6 00003DEF (01333)          ADDR N132-1
03DC8 00003DE4 (01334)          ADDR N063-1
03DCA 00003DFC (01335)          ADDR N074-1
03DCC 00003DF2 (01336)          ADDR N125-1
03DCE 00003DDD (01337)          ADDR N026-1
03DD0 00003DE9 (01338)          ADDR N177-1
               (01339)
               (01340) ;TABLE OF PTR-1 TO 2-BIT SEQUENCES
03DD2 00003F03 (01341) PTR2:    ADDR Q0-1
03DD4 00003DE8 (01342)          ADDR Q1-1
03DD6 00003DDF (01343)          ADDR Q2-1
03DD8 00003DF1 (01344)          ADDR Q3-1
               (01345)
               (01346) ;TABLE OF CODE SEQUENCES.
03DDA 000D     (01347) N151:    DATA 1 + TMBITS
03DDB 000D     (01348) P5:      DATA 1 + TMBITS
03DDC 000C     (01349) N045:    DATA 0 + TMBITS
03DDD 000D     (01350)          DATA 1 + TMBITS
03DDE 000C     (01351) N026:    DATA 0 + TMBITS
03DDF 000C     (01352) P2:      DATA 0 + TMBITS
03DE0 000D     (01353) Q2:      DATA 1 + TMBITS
03DE1 000C     (01354)          DATA 0 + TMBITS
03DE2 000D     (01355) N146:    DATA 1 + TMBITS
03DE3 000D     (01356) N114:    DATA 1 + TMBITS
03DE4 000C     (01357) N031:    DATA 0 + TMBITS
03DE5 000C     (01358) N063:    DATA 0 + TMBITS
03DE6 000D     (01359) P6:      DATA 1 + TMBITS
03DE7 000D     (01360)          DATA 1 + TMBITS
03DE8 000C     (01361) P1:      DATA 0 + TMBITS
03DE9 000C     (01362) Q1:      DATA 0 + TMBITS
03DEA 000D     (01363) N177:    DATA 1 + TMBITS
03DEB 000D     (01364) P7:      DATA 1 + TMBITS
03DEC 000D     (01365)          DATA 1 + TMBITS
03DED 000D     (01366)          DATA 1 + TMBITS
03DEE 000D     (01367)          DATA 1 + TMBITS
03DEF 000D     (01368)          DATA 1 + TMBITS
03DF0 000D     (01369) N132:    DATA 1 + TMBITS
03DF1 000C     (01370) P3:      DATA 0 + TMBITS
03DF2 000D     (01371) Q3:      DATA 1 + TMBITS
03DF3 000D     (01372) N125:    DATA 1 + TMBITS
03DF4 000C     (01373) N052:    DATA 0 + TMBITS
03DF5 000D     (01374)          DATA 1 + TMBITS
03DF6 000C     (01375)          DATA 0 + TMBITS
03DF7 000D     (01376)          DATA 1 + TMBITS
03DF8 000C     (01377)          DATA 0 + TMBITS
03DF9 000D     (01378) N103:    DATA 1 + TMBITS
03DFA 000C     (01379)          DATA 0 + TMBITS
03DFB 000C     (01380) N017:    DATA 0 + TMBITS
03DFC 000C     (01381)          DATA 0 + TMBITS
03DFD 000C     (01382) N074:    DATA 0 + TMBITS
```

```
PAGE   37:  [BBND]<DCA16>BBN160.MSD.2, 29-Dec-80 15:44:23, Ed: WOLF
            PROPES(AB), PROPAR(AB) -- ERROR-PROTECT AND BITSTREAM THE INTR FRAME.

      03DFE 000D       (01383)            DATA 1 + THBITS
      03DFF 000D       (01384) N160:      DATA 1 + THBITS
      03E00 000D       (01385)            DATA 1 + THBITS
      03E01 000D       (01386) P4:        DATA 1 + THBITS
      03E02 000C       (01387) N000:      DATA 0 + THBITS
      03E03 000C       (01388) P0:        DATA 0 + THBITS
      03E04 000C       (01389) Q0:        DATA 0 + THBITS
      03E05 000C       (01390)            DATA 0 + THBITS
      03E06 000C       (01391)            DATA 0 + THBITS
      03E07 000C       (01392)            DATA 0 + THBITS
      03E08 000C       (01393)            DATA 0 + THBITS
                       (01394)
```

```
                    "CORPAR(AB): UNBITSTREAM, ERROR-CORRECT, AND DECODE PARAMETERS
                    ;
                    ;CORPAR TAKES A FRAME OF BITSTREAM (RBITS BUFFER) FROM THE
                    ;  RECEIVER MODEM AND FORMS IT INTO PARAMETERS,
                    ;  DOING ERROR-CORRECTING DECODING FOR THE PROTECTED DATA BITS.  EACH
                    ;  UNBITSTREAMED AND CORRECTED VALUE IS THEN LOOKED UP IN THE
                    ;  APPROPRIATE DECODING TABLE, SO THAT THE OUTPUT (RSOURCE) BUFFER IS
                    ;  FULL-WORD FLOATING POINT VALUES READY FOR THE SYNTHESIS PROCESS.
                    ;CORPAR DOES NOT TOUCH THE RESIDUAL SAMPLES; THEY GET UNBITSTREAMED
                    ;  AND DECODED BY THE AP MODULES DECODA/DECODB.
                    ;CORPAR NO LONGER RUNS AT INTERRUPT LEVEL, BUT IS INVOKED VIA FCB.
                    ;
                    ;THIS ROUTINE IS ENTERED (VIA FCB) WITH R1 POINTING TO THE
                    ;  FCB0.  1(R1) IS THE BUFFER/FLAG POINTER OFFSET, EQUAL TO -2 FOR "A"
                    ;  OR 0 FOR "B".
                    ;REGISTER USAGE:
                    ;  R1 = PTR-1 TO THE INPUT (RBITS) BUFFER (USED BY POPXI'S)
                    ;  R2 = PTR TO THE OUTPUT (RSOURCE) BUFFER (USED IN PUSHNIL'S)
                    ;  R3-R5 = SCRATCH IN PARAMETER DECODING
                    ;  R6 - SCRATCH IN PARAM DECODING AND IN GET74, ETC.
                    ;  R7 - GET74, ETC. RETURN VALUES IN HERE
                    ;
                    ;THE FORMAT OF THE INPUT (RBITS) BUFFER IS IDENTICAL TO THE TBITS
                    ;  BUFFER, WHICH IS DESCRIBED ABOVE IN THE PRORES/PROPAR DOCUMENTATION.
                    ;  THE FORMAT OF THE OUTPUT (RSOURCE) BUFFER IS LIKE THAT OF THE
                    ;  TSINK BUFFER (ALSO DESCRIBED ABOVE), EXCEPT THAT EACH DATUM OCCUPIES
                    ;  A FULL-WORD INSTEAD OF A HALF-WORD, BECAUSE THESE ARE DECODED
                    ;  FLOATING POINT VALUES.
                    ;
                    ;POINTERS TO BUFFERS.  THE RBITS POINTERS ARE ALSO USED BY RMODEMINT.
03E09 0000                    EVEN
03E0A 0000039C                ADDR    RBTA              ;RBITS BUFFERS
03E0C 000005A6      RBTBPTR:  ADDR    RBTB
                    ;
03E0E 00000960                ADDR    RSRA+NRESID*W$  ;RSOURCE BUFFER PTRS TO PARAMETERS
03E10 00000B2C      RSRBPTR:  ADDR    RSRB+NRESID*W$  ;(STEPPING OVER THE RESIDUAL)
                    ;
                    ;
                              EVEN
03E12 F840FFCE      COPPAR:   MOVLM   SET+G1,SYS$FLGS   ;FOR SCOPE DISPLAY ...
03E14 F0220001                MOVMR   R2,1(R1)          ;PICK UP POINTER OFFSET
03E16 90943E0C                MOVIR   R1,@RBTBPTR(R2)   ;R1 GETS PTR TO RBITS BUFFER (POINTS TO
                    ;                                   ; SYNC BIT, SO IS PTR-1 TO THE DATA)
03E18 9C100100                ADDIR   R1,NRESID*2*H$    ;STEP OVER NRESID 2-BIT DATA
03E1A 90A43E10                MOVIR   P2,@RSRBPTR(R2)   ;R2 GETS PTR TO RSOURCE BUFFER PARAMS
03E1C 86003EA0                CALL    R0,GET74          ;GET 4 HIGH BITS OF GAIN
03E1E 3A72                    LLS     R7,2              ;SHIFT THEM TO FINAL POSITION
03E1F 405E                    MOVRR   R5,R7             ;AND SAVE THEM IN R5
                    ;
03E20 86003EA0                CALL    R0,GET74          ;GET LO 2 GAIN, DG1 (2 BITS)
03E22 504E0006                MOVKR   R4,R7,C10         ;STASH DG1 IN R4
03E24 3C72                    LRS     R7,2              ;SHIFT B10 GAIN INTO FINAL POSITION
03E25 465E                    IORRR   R5,R7             ; AND ADD TO B5432
```

```
03E26 C62A9870 (01448)        PUSHMIL R2,DVLG(R5)     ;DECODE AND STORE GAIN
03E28 C62B98F0 (01449)        PUSHMIL R2,DVLDG(R4)    ;DECODE AND STORE DG1
               (01450) ;
03E2A 86003EA0 (01451)        CALL    R0,GET74        ;GET DG2, DG3
03E2C 566F0006 (01452)        MOVKR   R6,R7,C10       ;SAVE DG3 IN R6
03E2E 3C72     (01453)        LRS     R7,2            ;SHIFT DG2 INTO POSITION
03E2F C62F98F0 (01454)        PUSHMIL R2,DVLDG(R7)    ;DECODE AND STORE DG2
03E31 C62C98F0 (01455)        PUSHMIL R2,DVLDG(R6)    ;DECODE AND STORE DG3
               (01456) ;
03E33 86003EA0 (01457)        CALL    R0,GET74        ;GET HI 4 BITS OF PITCH
03E35 3A73     (01458)        LLS     R7,3            ;SHIFT INTO FINAL POSITION
03E36 405F     (01459)        MOVRR   R5,R7           ; AND SAVE IN R5
03E37 86003ED6 (01460)        CALL    R0,GET1         ;GET B0 C1
03E39 404F     (01461)        MOVRR   R4,R7           ;SAVE IT IN R4
03E3A 86003EA0 (01462)        CALL    R0,GET74        ;GET B2 C1, LO 3 BITS OF PITCH
03E3C 565F000E (01463)        IORKR   R5,R7,C210      ;MOVE LO 3 OF PITCH IN WITH HI 4
03E3E 9C50001C (01464)        ADDIR   R5,14*2         ;ADD 14 TO "DECODE" (LSHIFTED 1, SO 2*)
03E40 3A56     (01465)        LLS     R5,7-1          ;MAKE IT INTO UNNORM FLT PT IN R5,R6
03E41 90600042 (01466)        MOVIR   R6,$42          ;EXPONENT
03E43 84540000 (01467)        MOVRML  R5,0(R2)        ;STORE R5,R6
03E45 2622     (01468)        INCR    R2,2            ;STEP OUTPUT POINTER
03E46 3C71     (01469)        LRS     R7,1            ;SHIFT B2 OF C1 INTO B2 POSITION
03E47 564F0000 (01470)        IORKR   R4,R7,C2        ;R4 NOW HAS B2, B0 OF C1
               (01471) ;
03E49 86003ED6 (01472)        CALL    R0,GET1         ;GET B0 OF C2
03E4B 403F     (01473)        MOVRR   R3,R7           ;SAVE IT IN R3
               (01474) ;
03E4C 86003EA0 (01475)        CALL    R0,GET74        ;GET B321 OF C2, B1 OF C1
03E4E 563F001C (01476)        IORKR   R3,R7,C321      ;ADD B321 OF C2 TO B0
03E50 4C7F     (01477)        ADDRR   R7,R7           ;LEFTSHIFT TO POSITION B1 OF C1
03E51 564F0004 (01478)        IORKR   R4,R7,C1        ;OR IT IN WITH B2, B0 OF C1
03E53 C62B9A00 (01479)        PUSHMIL R2,DVLC1(R4)    ;DECODE AND STORE C1
03E55 C6269A10 (01480)        PUSHMIL R2,DVLC2(R3)    ;DECODE AND STORE C2
               (01481) ;
03E57 86003ED6 (01482)        CALL    R0,GET1         ;GET B0 OF C3
03E59 405F     (01483)        MOVRR   R5,R7           ;SAVE IT IN R5
               (01484) ;
03E5A 86003ED6 (01485)        CALL    R0,GET1         ;GET B0 K1
03E5C 404E     (01486)        MOVRR   R4,R7           ;SAVE IT IN R4
               (01487) ;
03E5D 86003EA0 (01488)        CALL    R0,GET74        ;GET B54 K1, B21 C3
03E5F 4C7E     (01489)        ADDRR   R7,R7           ;LEFTSHIFT TO POSITION B21 OF C3
03E60 565F000C (01490)        IORKR   R5,R7,C21       ;PUT B21 C3 IN WITH B0
03E62 C62A9A00 (01491)        PUSHMIL R2,DVLC3(R5)    ;DECODE AND STORE C3
03E64 4C7E     (01492)        ADDRR   R7,R7           ;LEFTSHIFT AGAIN TO POSITION B54 OF K1
03E65 564F0060 (01493)        IORKR   R4,R7,C54       ;SAVE THEM IN R4 WITH B0
               (01494) ;
03E67 86003ED6 (01495)        CALL    R0,GET1         ;GET B0 K2
03E69 405F     (01496)        MOVRR   R5,R7           ;STASH IT IN R5
               (01497) ;
03E6A 86003EA0 (01498)        CALL    R0,GET74        ;GET 4 MSB'S OF K2
03E6C 4C7E     (01499)        ADDRR   R7,R7           ;LEFT SHIFT 1 TO POSITION THEM
03E6D 465E     (01500)        IORRR   R5,R7           ;ADD B4321 K2 TO B0 IN R5
```

```
                  ;
03E6E 86003ED6            CALL     R0,GET1          ;GET B0 OF K3
03E70 403R                MOVRR    R3,R7            ;STASH IT IN R3
                  ;
03E71 86003EA0            CALL     R0,GET74         ;GET B3 K3, B321 K1
03E73 563F0010            IORKR    R3,R7,C3         ;PUT B3 K3 IN R3 ALONG WITH B0
03E75 4C7R                ADDRR    R7,R7            ;LEFT SHIFT 1 TO POSITION B321 K1
03E76 564E001C            IORKR    R4,R7,C321       ;PUT B321 K1 IN R4 WITH B54 AND B0
03E78 C6289A30            PUSHMIL  R2,DVLK1(R4)     ;DECODE AND STORE K1
03E7A C62A9A00            PUSHMIL  R2,DVLK2(R5)     ;DECODE AND STORE K2
                  ;
03E7C 86003EA0            CALL     R0,GET74         ;GET B32 K4, B21 K3
03E7E 505E0010            MOVKR    R5,R7,C32        ;SAVE B32 K4 IN R5
03E80 4C7F                ADDRR    R7,R7            ;LEFTSHIFT 1 TO POSITION B21 K3
03E81 563E000C            IORKR    R3,R7,C21        ;PUT B21 K3 IN R3 ALONG WITH B3 AND B0
03E83 C6269AF0            PUSHMIL  R2,DVLK3(R3)     ;DECODE AND STORE K3
                  ;
03E85 86003ECC            CALL     R0,GET2          ;GET B10 K4
03E87 565F0006            IORKR    R5,R7,C10        ;PUT THEM IN R5 ALONG WITH B32
03E89 C62A9B10            PUSHMIL  R2,DVLK4(R5)     ;DECODE AND STORE K4
                  ;
03E8B 86003ECC            CALL     R0,GET2          ;GET B10 K5
03E8D 405E                MOVRR    R5,R7            ;SAVE IT IN R5
03E8E 86003ECC            CALL     R0,GET2          ;GET B10 K6
03E90 404F                MOVRR    R4,R7            ;SAVE IT IN R4
03E91 86003EA0            CALL     R0,GET74         ;GET B32 K6, B32 K5
03E93 564E0010            IORKR    R4,R7,C32        ;PUT B32 K6 IN R4 ALONG WITH B10 K6
03E95 3A72                LLS      R7,2             ;LEFTSHIFT 2 TO POSITION B32 K5
03E96 565F0010            IORKR    R5,R7,C32        ;PUT B32 K5 IN R5 ALONG WITH B10 K5
03E98 C62A9B30            PUSHMIL  R2,DVLK5(R5)     ;DECODE AND STORE K5
03E9A C6289B50            PUSHMIL  R2,DVLK6(R4)     ;DECODE AND STORE K6
                  ;
03E9C F00HFFCE            MOVLM    CLR+GI,SYS$FLGS
03E9E 0E70                RETURN
                  ;
                  ;THESE UNBITSTREAMING ROUTINES RETURN THEIR VALUES IN R7 LEFT
                  ;  SHIFTED ONE PLACE, SO THAT THEY CAN ACT AS FULL-WORD OFFSETS INTO
                  ;  THE DECODING TABLES.
                  ;
                  ;PULL IN 7 SUCCESSIVE WORDS FROM THE BITSTREAM AND FORM THEIR DATA
                  ;  BITS INTO A 7-BIT VALUE.  THIS VALUE IS USED AS AN INDEX INTO
                  ;  CORTAB TO YIELD AN ERROR-CORRECTED 4-BIT VALUE.  IF RNCOR IS NZ,
                  ;  OMIT THE ERROR-CORRECTION AND RETURN ONLY THE 4 INFORMATION BITS.
                  ;R1 = PTR-1 TO BITSTREAM
                  ;R6 = SCRATCH
                  ;R7 = HOLDS RETURNED VALUE
                  ;THIS CAN BE CALLED WITH CALL R0,GET74.
03E9F 0R00                EVEN
03EA0 3P1C        GET74:  POPXI    R1,R6            ;GET MODEM WORD INTO R6
03EA1 0R00                EVEN
03EA2 507C0001            MOVKR    R7,R6,1          ;EXTRACT DATA BIT TO R7
03EA4 4C7F                ADDRR    R7,R7            ;SHIFT LEFT 1 FASTER THAN LLS
03EA5 3R1C                POPXI    R1,R6
```

```
03EA6 567C0001 (01554)         IORKR   R7,R6,1
03EA8 4C7E     (01555)         ADDRR   R7,R7
03EA9 301C     (01556)         POPXI   R1,R6
03EAA 567C0001 (01557)         IORKR   R7,R6,1
03EAC 4C7E     (01558)         ADDRR   R7,R7
03EAD 301C     (01559)         POPXI   R1,R6
03EAE 567C0001 (01560)         IORKR   R7,R6,1
03EB0 4C7E     (01561)         ADDRR   R7,R7
03EB1 301C     (01562)         POPXI   R1,R6
03EB2 567C0001 (01563)         IORKR   R7,R6,1
03EB4 4C7E     (01564)         ADDRR   R7,R7
03EB5 301C     (01565)         POPXI   R1,R6
03EB6 567C0001 (01566)         IORKR   R7,R6,1
03EB8 4C7E     (01567)         ADDRR   R7,R7
03EB9 301C     (01568)         POPXI   R1,R6
03EBA 567C0001 (01569)         IORKR   R7,R6,1
03EBC E2000570 (01570)         CMPNZ   RNCOR           ;DO WE ERROR-CORRECT?
03EBE 8110 3EC4 (01571)        JMP     NOCORR,NEZ      ;JUMP IF NO
03EC0 F07R3ED0 (01572) ERCORR: MOVMR   R7,CORTAB(R7)   ;LOOK UP CORRECTED 4-BIT VALUE
03EC2 0E70     (01573)         RETURN
               (01574) ;
03EC3 0000     (01575)         EVEN
03EC4 506E0010 (01576) NOCORR: MOVKR   R6,R7,C3        ;NO ERROR CORRECTION: GET B3 INFO BIT
03EC6 4C7E     (01577)         ADDRR   R7,R7           ;SHIFT THE OTHERS
03EC7 0000     (01578)         EVEN
03EC8 566E000E (01579)         IORKR   R6,R7,C210      ;EXTRACT AND ADD B2,B1,B0 INFO BITS
03ECA 407C     (01580)         MOVRR   R7,R6
03ECB 0E70     (01581)         RETURN
               (01582) ;
               (01583) ;GET1 AND GET2 UNBITSTREAM 1 AND 2 BITS, ALSO RETURNING RESULT
               (01584) ;  LEFT-SHIFTED ONE BIT.
               (01585)         EVEN
03ECC 301C     (01586) GET2:   POPXI   R1,R6
03ECD 0000     (01587)         EVEN
03ECE 507C0001 (01588)         MOVKR   R7,R6,1
03ED0 4C7E     (01589)         ADDRR   R7,R7
03ED1 301C     (01590) GT1:    POPXI   R1,R6
03ED2 567C0001 (01591)         IORKR   R7,R6,1
03ED4 4C7E     (01592)         ADDRR   R7,R7
03ED5 0E70     (01593)         RETURN
               (01594) ;
               (01595)         EVEN
03ED6 0D70     (01596) GET1:   CLR     R7
03ED7 2107     (01597)         NOP     GT1
               (01598) ;
               (01599) ;HAMMING 7/4 CODE CORRECTION TABLE.  INDEX TABLE WITH 7-BIT RECEIVED
               (01600) ;  CODEWORD; TABLE VALUE IS THE CORRECTED 4-BIT VALUE,
               (01601) ;  LEFT SHIFTED 1 BIT (TO BE USED AS A FULL-WORD OFFSET IN THE DECODING
               (01602) ;  TABLES).
               (01603) CORTAB:
03ED0 0000     (01604)         DATA  O'00'*2    ;RCVD 000
03ED9 0000     (01605)         DATA  O'00'*2    ;RCVD 001
03EDA 0000     (01606)         DATA  O'00'*2    ;RCVD 002
```

```
03EDB 0006     (01607)     DATA  0'03'*2    ;RCVD 003
03EDC 0008     (01608)     DATA  0'04'*2    ;RCVD 004
03EDD 000A     (01609)     DATA  0'05'*2    ;RCVD 005
03EDE 001C     (01610)     DATA  0'16'*2    ;RCVD 006
03EDF 000E     (01611)     DATA  0'07'*2    ;RCVD 007
03EE0 0000     (01612)     DATA  0'00'*2    ;RCVD 010
03EE1 0012     (01613)     DATA  0'11'*2    ;RCVD 011
03EE2 0004     (01614)     DATA  0'02'*2    ;RCVD 012
03EE3 000E     (01615)     DATA  0'07'*2    ;RCVD 013
03EE4 0008     (01616)     DATA  0'04'*2    ;RCVD 014
03EE5 000E     (01617)     DATA  0'07'*2    ;RCVD 015
03EE6 000E     (01618)     DATA  0'07'*2    ;RCVD 016
03EE7 000E     (01619)     DATA  0'07'*2    ;RCVD 017
03EE8 0000     (01620)     DATA  0'00'*2    ;RCVD 020
03EE9 0012     (01621)     DATA  0'11'*2    ;RCVD 021
03EEA 001C     (01622)     DATA  0'16'*2    ;RCVD 022
03EEB 0016     (01623)     DATA  0'13'*2    ;RCVD 023
03EEC 001C     (01624)     DATA  0'16'*2    ;RCVD 024
03EED 001A     (01625)     DATA  0'15'*2    ;RCVD 025
03EEE 001C     (01626)     DATA  0'16'*2    ;RCVD 026
03EEF 001C     (01627)     DATA  0'16'*2    ;RCVD 027
03EF0 0012     (01628)     DATA  0'11'*2    ;RCVD 030
03EF1 0012     (01629)     DATA  0'11'*2    ;RCVD 031
03EF2 0014     (01630)     DATA  0'12'*2    ;RCVD 032
03EF3 0012     (01631)     DATA  0'11'*2    ;RCVD 033
03EF4 0018     (01632)     DATA  0'14'*2    ;RCVD 034
03EF5 0012     (01633)     DATA  0'11'*2    ;RCVD 035
03EF6 001C     (01634)     DATA  0'16'*2    ;RCVD 036
03EF7 000E     (01635)     DATA  0'07'*2    ;RCVD 037
03EF8 0000     (01636)     DATA  0'00'*2    ;RCVD 040
03EF9 000A     (01637)     DATA  0'05'*2    ;RCVD 041
03EFA 0004     (01638)     DATA  0'02'*2    ;RCVD 042
03EFB 0016     (01639)     DATA  0'13'*2    ;RCVD 043
03EFC 000A     (01640)     DATA  0'05'*2    ;RCVD 044
03EFD 000A     (01641)     DATA  0'05'*2    ;RCVD 045
03EFE 000C     (01642)     DATA  0'06'*2    ;RCVD 046
03EFF 000A     (01643)     DATA  0'05'*2    ;RCVD 047
03F00 0004     (01644)     DATA  0'02'*2    ;RCVD 050
03F01 0002     (01645)     DATA  0'01'*2    ;RCVD 051
03F02 0004     (01646)     DATA  0'02'*2    ;RCVD 052
03F03 0004     (01647)     DATA  0'02'*2    ;RCVD 053
03F04 0018     (01648)     DATA  0'14'*2    ;RCVD 054
03F05 000A     (01649)     DATA  0'05'*2    ;RCVD 055
03F06 0004     (01650)     DATA  0'02'*2    ;RCVD 056
03F07 000E     (01651)     DATA  0'07'*2    ;RCVD 057
03F08 0010     (01652)     DATA  0'10'*2    ;RCVD 060
03F09 0016     (01653)     DATA  0'13'*2    ;RCVD 061
03F0A 0016     (01654)     DATA  0'13'*2    ;RCVD 062
03F0B 0016     (01655)     DATA  0'13'*2    ;RCVD 063
03F0C 0018     (01656)     DATA  0'14'*2    ;RCVD 064
03F0D 000A     (01657)     DATA  0'05'*2    ;RCVD 065
03F0E 001C     (01658)     DATA  0'16'*2    ;RCVD 066
03F0F 0016     (01659)     DATA  0'13'*2    ;RCVD 067
```

```
03F10 0018      (01660)      DATA  D'14'*2     ;RCVD 070
03F11 0012      (01661)      DATA  D'11'*2     ;RCVD 071
03F12 0004      (01662)      DATA  D'02'*2     ;RCVD 072
03F13 0016      (01663)      DATA  D'13'*2     ;RCVD 073
03F14 0018      (01664)      DATA  D'14'*2     ;RCVD 074
03F15 0018      (01665)      DATA  D'14'*2     ;RCVD 075
03F16 0018      (01666)      DATA  D'14'*2     ;RCVD 076
03F17 001E      (01667)      DATA  D'17'*2     ;RCVD 077
03F18 0000      (01668)      DATA  D'00'*2     ;RCVD 100
03F19 0006      (01669)      DATA  D'03'*2     ;RCVD 101
03F1A 0006      (01670)      DATA  D'03'*2     ;RCVD 102
03F1B 0006      (01671)      DATA  D'03'*2     ;RCVD 103
03F1C 0008      (01672)      DATA  D'04'*2     ;RCVD 104
03F1D 001A      (01673)      DATA  D'15'*2     ;RCVD 105
03F1E 000C      (01674)      DATA  D'06'*2     ;RCVD 106
03F1F 0006      (01675)      DATA  D'03'*2     ;RCVD 107
03F20 0008      (01676)      DATA  D'04'*2     ;RCVD 110
03F21 0002      (01677)      DATA  D'01'*2     ;RCVD 111
03F22 0014      (01678)      DATA  D'12'*2     ;RCVD 112
03F23 0006      (01679)      DATA  D'03'*2     ;RCVD 113
03F24 0008      (01680)      DATA  D'04'*2     ;RCVD 114
03F25 0008      (01681)      DATA  D'04'*2     ;RCVD 115
03F26 0008      (01682)      DATA  D'04'*2     ;RCVD 116
03F27 000E      (01683)      DATA  D'07'*2     ;RCVD 117
03F28 0010      (01684)      DATA  D'10'*2     ;RCVD 120
03F29 001A      (01685)      DATA  D'15'*2     ;RCVD 121
03F2A 0014      (01686)      DATA  D'12'*2     ;RCVD 122
03F2B 0006      (01687)      DATA  D'03'*2     ;RCVD 123
03F2C 001A      (01688)      DATA  D'15'*2     ;RCVD 124
03F2D 001A      (01689)      DATA  D'15'*2     ;RCVD 125
03F2E 001C      (01690)      DATA  D'16'*2     ;RCVD 126
03F2F 001A      (01691)      DATA  D'15'*2     ;RCVD 127
03F30 0014      (01692)      DATA  D'12'*2     ;RCVD 130
03F31 0012      (01693)      DATA  D'11'*2     ;RCVD 131
03F32 0014      (01694)      DATA  D'12'*2     ;RCVD 132
03F33 0014      (01695)      DATA  D'12'*2     ;RCVD 133
03F34 0008      (01696)      DATA  D'04'*2     ;RCVD 134
03F35 001A      (01697)      DATA  D'15'*2     ;RCVD 135
03F36 0014      (01698)      DATA  D'12'*2     ;RCVD 136
03F37 001E      (01699)      DATA  D'17'*2     ;RCVD 137
03F38 0010      (01700)      DATA  D'10'*2     ;RCVD 140
03F39 0002      (01701)      DATA  D'01'*2     ;RCVD 141
03F3A 000C      (01702)      DATA  D'06'*2     ;RCVD 142
03F3B 0006      (01703)      DATA  D'03'*2     ;RCVD 143
03F3C 000C      (01704)      DATA  D'06'*2     ;RCVD 144
03F3D 000A      (01705)      DATA  D'05'*2     ;RCVD 145
03F3E 000C      (01706)      DATA  D'06'*2     ;RCVD 146
03F3F 000C      (01707)      DATA  D'06'*2     ;RCVD 147
03F40 0002      (01708)      DATA  D'01'*2     ;RCVD 150
03F41 0002      (01709)      DATA  D'01'*2     ;RCVD 151
03F42 0004      (01710)      DATA  D'02'*2     ;RCVD 152
03F43 0002      (01711)      DATA  D'01'*2     ;RCVD 153
03F44 0008      (01712)      DATA  D'04'*2     ;RCVD 154
```

COMPAR(AB): UNBITSTREAM, ERROR-CORRECT, AND DECODE PARAMETERS

```
03F45 0002        (01713)        DATA  0'01'*2      ;RCVD 155
03F46 000C        (01714)        DATA  0'06'*2      ;RCVD 156
03F47 001E        (01715)        DATA  0'17'*2      ;RCVD 157
03F48 0010        (01716)        DATA  0'10'*2      ;RCVD 160
03F49 0010        (01717)        DATA  0'10'*2      ;RCVD 161
03F4A 0010        (01718)        DATA  0'10'*2      ;RCVD 162
03F4B 0016        (01719)        DATA  0'13'*2      ;RCVD 163
03F4C 0010        (01720)        DATA  0'10'*2      ;RCVD 164
03F4D 001A        (01721)        DATA  0'15'*2      ;RCVD 165
03F4E 000C        (01722)        DATA  0'06'*2      ;RCVD 166
03F4F 001E        (01723)        DATA  0'17'*2      ;RCVD 167
03F50 0010        (01724)        DATA  0'10'*2      ;RCVD 170
03F51 0002        (01725)        DATA  0'01'*2      ;RCVD 171
03F52 0014        (01726)        DATA  0'12'*2      ;RCVD 172
03F53 001E        (01727)        DATA  0'17'*2      ;RCVD 173
03F54 0018        (01728)        DATA  0'14'*2      ;RCVD 174
03F55 001E        (01729)        DATA  0'17'*2      ;RCVD 175
03F56 001E        (01730)        DATA  0'17'*2      ;RCVD 176
03F57 001E        (01731)        DATA  0'17'*2      ;RCVD 177
                  (01732)
```

```
                        (01733) * APU3-DECOD()           DECODE RESIDUAL
                        (01734) ;  J. WOLF, 10/21/80
                        (01735) ;
                        (01736) ;  BINDS TO APS3-DECODA OR APS3-DECODB
                        (01737) ;
                        (01738) ;UNBITSTREAMS AND DECODES THE RECEIVED RESIDUAL BITS FROM AN "RBITS"
                        (01739) ;  BUFFER (RBTA OR RBTB) TO AN "RSOURCE" BUFFER (RSRA OR RSRB).
                        (01740) ;        RSRX(N) = DVLUF(RBTX(N+2),RBTX(N+1))
                        (01741) ;  WE PULL IN 2 HALFWORDS FROM THE RBTX BUFFER, THE FIRST OF WHICH HAS
                        (01742) ;    THE RESIDUAL LSB IN ITS RIGHTMOST BIT (B16) AND THE SECOND OF WHICH
                        (01743) ;    HAS THE MSB.  WE SHIFT THOSE BITS TO BIT 7, THEN XLSBT THEM
                        (01744) ;    INTO T1 AND T2 RESPECTIVELY, THEN DO LOGIC ON THE T-BITS TO
                        (01745) ;    SELECT THE PROPER OUTPUT VALUE FROM THE 4-LONG DVLUF TABLE.
                        (01746) ;  DVLUF IS THE (LENGTH=4) DECODING TABLE FOR THE 2-BIT RESIDUAL DATA,
                        (01747) ;    WHICH USES THE FOLDED BINARY CODE.
                        (01748) ;
                        (01749) ;INPUT SEQUENCE:
                        (01750) ;  (2**-9), DVLUF(0),...,DVLUF(3),
                        (01751) ;  RBTX(1), RBTX(2), ..., RBTX(2*NRESID), [FI]
                        (01752) ;OUTPUT SEQUENCE:
                        (01753) ;  RSRX(0), RSRX(1), ..., RSRX(NRESID-1), [EO]
                        (01754) ;
                        (01755) ;DURATION:
                        (01756) ;  ~10 APU CYCLES = ~1.0 USEC PER RESIDUAL SAMPLE
                        (01757) ;   3 MEM REFS    = ~1.5 USEC (BUS 1) PER RESIDUAL SAMPLE
                        (01758) ;
                        (01759) ;APU REGISTER USAGE:
                        (01760) ;         LEFT              RIGHT
                        (01761) ; A0      DVLUF(0)            -
                        (01762) ; A1      DVLUF(1)            -
                        (01763) ; A2      DVLUF(2)            -
                        (01764) ; A3      DVLUF(3)            -
                        (01765) ; A4:     SHIFTED LSB       SHIFTED MSB
                        (01766) ; M0:     2**-9             2**-9
                        (01767) ; M4:     LSB WORD          MSB WORD
                        (01768) ;
                        (01769)           EVEN
    03F58 0000          (01770)           DATA DCDU$SA     ;APU START ADDRESS
    03F59 001F          (01771)           DATA DCDU$SZ     ;APU MODULE SIZE
                        (01772) ;
                        (01773) DCDU:     BEGIN APU(DCDU)
A00 03F5A 20532053      (01774) DCDU$SA:  SET(UN)                       ;WE'LL BE MULTIPLYING UNNORM NUMBERS
A01 03F5C 00E000E0      (01775)           MOV(IQA,M0)                   ;2**-9 \ 2**-9
A02 03F5E 00F00000      (01776)           MOV(IQA,A0) \ NOP             ;DVLUF(0)
A03 03F60 00F10000      (01777)           MOV(IQA,A1) \ NOP             ;DVLUF(1)
A04 03F62 00F20000      (01778)           MOV(IQA,A2) \ NOP             ;DVLUF(2)
A05 03F64 00F30000      (01779)           MOV(IQA,A3) \ NOP             ;DVLUF(3)
A06 03F66 00EC0000      (01780)           MOV(IQA,M4) \ NOP             ;PICK UP WORD CONTAINING LSB
A07 03F68 000000EC      (01781)           NOP \ MOV(IQA,M4)             ;... AND THE MSB WORD TOO
A08 03F6A 04000400      (01782)           MUL(M0,M4)                    ;SHIFT BOTH LSB AND MSB 9 PLACES TO THE
A09 03F6C 00840084      (01783)           MOV(P,A4)                     ; RIGHT (TO D7, THE LSB OF THE MANTISSA)
                        (01784) ;MAIN LOOP
A0A 03F6E 3A003A00      (01785) 01:       ALIGN(A4)                     ;BY MULT. BY (2**-9) AND THEN ALIGNING.
```

```
A0B 03F70 00EC0000 (01786)            MOV(IQA,M4) \ NOP       ;(PICK UP THE NEXT LSB)
A0C 03F72 1A941A94 (01787)            MOV(A4),XLSBT(A4)       ;MOVE THE DATA BITS INTO T1, T2
A0D 03F74 000000EC (01788)            NOP \ MOV(IQA,M4)       ;(PICK UP THE NEXT MSB)
A0E 03F76 84000400 (01789)            MUL(M0,M4)              ;(START THE NEXT MULTIPLY)
A0F 03F78 00000000 (01790)            MOV(R,NULL)             ;WAIT FOR XLSBT TO FINISH
A10 03F7A 911F0016 (01791)            JUMPS(DCD$1X,T2)        ;JUMP IF MSB=1
A11 03F7C 911E0014 (01792) DCD$0X:    JUMPS(DCD$01,T1)        ;MSB=0; JUMP IF LSB=1
A12 03F7E 02000000 (01793) DCD$00:    R(A0) \ NOP             ;DATUM=00; RETURN DVLUF(0)
A13 03F80 1000001A (01794)            JUMP(DCD$OUT)
                   (01795) ;
A14 03F82 02200000 (01796) DCD$01:    R(A1) \ NOP             ;DATUM=01: DVLUF(1)
A15 03F84 1000001A (01797)            JUMP(DCD$OUT)
                   (01798) ;
A16 03F86 911E0019 (01799) DCD$1X:    JUMPS(DCD$11,T1)        ;MSB=1; JUMP IF LSB=1
A17 03F88 02400000 (01800) DCD$10:    R(A2) \ NOP             ;DATUM=10: DVLUF(2)
A18 03F8A 1000001A (01801)            JUMP(DCD$OUT)
                   (01802) ;
A19 03F8C 02600000 (01803) DCD$11:    R(A3) \ NOP             ;DATUM=11: DVLUF(3)
A1A 03F8E 008400B4 (01804) DCD$OUT:   MOV(P,A4)
A1B 03F90 009C0000 (01805)            MOV(R,OQ) \ NOP         ;OUTPUT DECODED VALUE
A1C 03F92 901C000A (01806)            JUMPC($1,EO)            ;LOOP IF WE'VE NOT FINISHED OUTPUT
A1D 03F94 20322032 (01807)            CLEAR(RA)
A1E 03F96 10000000 (01808)            JUMP(0)
          0000001F (01809) DCDU$SZ  = $A - DCDU$SA
    03F98          (01810)            END
```

```
                         ' APS3-DECODA() APS PROGRAM FOR DECODA
                         ;
                         ;  J. WOLF, 10/21/80
                         ;
                         ;  BINDS TO APU3-DECOD
                         ;
                         ;  APS PROGRAM FOR DECODING THE RESIDUAL FOR "A" BUFFERS (RBTA TO RSRA).
                         ;    BECAUSE WE CAN'T SPARE BID'S FOR THE RBTX BUFFERS, WE HAVE 2
                         ;    VERSIONS OF THE DECODE MODULE, ONE WITH (RBTA,RSRA) "HARD BOUND"
                         ;    IN, AND ONE WITH (RBTB,RSRB).
                         ;
                         ; HEADER BLOCK
                         ;
                                   EVEN
    03F98 00000000                 ADDR 0           ;NO CONSTRUCTED INSTRUCTIONS
    03F9A 00000000                 ADDR 0           ;NO SCALAR TABLE ENTRIES
    03F9C 0000                     DATA 0
    03F9D 002A                     DATA DCDSA$Z     ;MODULE SIZE
    03F9E 00000000                 ADDR 0           ;NO CHAIN ANCHOR
                                   EVEN
                         ;
                         ;INPUT PROGRAM:
                         ;  BR0: RBTA(N)
                         ;  BR1: LOOP COUNTER (STARTS AT NRESID-1)
                         ;
                         DCDSA:
                         DCDSAI: BEGIN APS(DCDSA)
A00 03FA0 002A0E40                 JSN(DCDSAO,P2)           ;SET AND START OUTPUT PROGRAM
A01 03FA2 02300030                 SET(RO)
A02 03FA4 04C23FCA                 LOAD(BR0,TWOM9(1),TF)    ;GEN ADR FOR (2**-9)
A03 03FA6 06C29C00                 LOAD(BR0,DVLUF(1),TF)    ;GEN DVLUF(0)
A04 03FA8 080A0002                 ADD(BR0,W$,TF)           ;DVLUF(1)
A05 03FAA 0A0A0002                 ADD(BR0,W$,TF)           ;DVLUF(2)
A06 03FAC 0C0A0002                 ADD(BR0,W$,TF)           ;DVLUF(3)
                         ;
A07 03FAE 0E42B39C                 LOAD(BR0,RBTA(1))        ;BR0 POINTS TO INPUT DATA - H$
A08 03FB0 105000D7                 LOAD(BR1,NRESID-1)       ;INIT BR1 TO COUNT NRESID SAMPLES
                         ;MAIN LOOP
A09 03FB2 130A0001       01:       ADD(BR0,H$,TE)           ;GEN ADR FOR LSB WORD
A0A 03FB4 150A0001                 ADD(BR0,H$,TE)           ;... AND MSB WORD
A0B 03FB6 161909B1                 SUBL(BR1,1),JUMPP(01)    ;LOOP IF MORE TO DO
A0C 03FB8 18200031                 CLFAR(RI)
A0D 03FBA 1A000020                 NOP(0)
                         ;
                         ;OUTPUT PROGRAM:
                         ;  BW0: RSRA(N)
                         ;  BW1: LOOP COUNTER (INIT TO NRESID-1)
                         ;
A0E 03FBC 1C300032       DCDSAO: SET(RA)                    ;START APU
A0F 03FBE 1E42B7AE                 LOAD(BW0,RSRA-W$(1))     ;INIT BW0 TO POINT TO OUTPUT BUFFER-W$
A10 03FC0 205000D7                 LOAD(BW1,NRESID-1)       ;INIT LOOP COUNTER
                         ;MAIN LOOP:
A11 03FC2 220A0002       02:       ADD(BW0,W$,TF)           ;GEN OUTPUT ADR
```

```
A12 03FC4 24111101 (01864)          SUBL(BW1,1), JUMPP(02)
A13 03FC6 26200030 (01865)          CLEAR(RO)
A14 03FC8 20000020 (01866)          NOP(0)
    03FCA              (01867)          END
          0000002A (01868) DCDSASZ = $L - DCDSAI          ;COMPUTE BUS 1 MODULE SIZE
                   (01869) ;
    03FCA 4000     (01870) TWON9:  DATA $4000,$003E       ;(2**-9) FOR DECODA, DECODB
    03FCB 003E
```

```
PAGE    49:  [BBNO]<DCA16>BBN16U.MSU.2, 29-Dec-80 15:44:23, Ed: WOLF
              APS3-DECODB()        APS PROGRAM FOR DECODB

                             (01871) ' APS3-DECODB() APS PROGRAM FOR DECODB
                             (01872) ;
                             (01873) ;  BINDS TO APU3-DECOD
                             (01874) ;
                             (01875) ;  SAME AS APS3-DECUDA, EXCEPT FOR "B" BUFFERS INSTEAD OF "A".
                             (01876) ;
                             (01877) ; HEADER BLOCK
                             (01878) ;
                             (01879)         EVEN
        03FCC 00000000 (01880)         ADDR 0          ;NO CONSTRUCTED INSTRUCTIONS
        03FCE 00000000 (01881)         ADDR 0          ;NO SCALAR TABLE ENTRIES
        03FD0 0000     (01882)         DATA 0
        03FD1 002A     (01883)         DATA DCDSB$Z    ;MODULE SIZE
        03FD2 00000000 (01884)         ADDR 0          ;NO CHAIN ANCHOR
                             (01885)         EVEN
                             (01886) ;
                             (01887) ;INPUT PROGRAM:
                             (01888) ;  BR0: RBTB(N)
                             (01889) ;  BR1: LOOP COUNTER (STARTS AT NRESID-1)
                             (01890) ;
                             (01891) DCDSB:
                             (01892) DCDSBI: BEGIN APS(DCDSB)
A00 03FD4 00200E40 (01893)         JSN(DCDSBO,P2)          ;SET AND START OUTPUT PROGRAM
A01 03FD6 02300030 (01894)         SET(RQ)
A02 03FD8 04C23FCA (01895)         LOAD(BR0,TWOM9(1),TF)   ;GEN ADR FOR (2**-9)
A03 03FDA 06C29C00 (01896)         LOAD(BR0,DVLUF(1),TF)   ;GEN DVLUF(0)
A04 03FDC 008A0002 (01897)         ADD(BR0,W$,TF)          ;DVLUF(1)
A05 03FDE 0A0A0002 (01898)         ADD(BR0,W$,TF)          ;DVLUF(2)
A06 03FE0 0C8A0002 (01899)         ADD(BR0,W$,TF)          ;DVLUF(3)
                             (01900) ;
A07 03FF2 0E42B5A6 (01901)         LOAD(BR0,RBTB(1))       ;BR0 POINTS TO INPUT DATA - H$
A08 03FE4 105000D7 (01902)         LOAD(BR1,NRESID-1)      ;INIT BR1 TO COUNT NRESID SAMPLES
                             (01903) ;MAIN LOOP
A09 03FF6 130A0001 (01904) 01:     ADD(BR0,H$,TE)          ;GEN ADR FOR LSB WORD
A0A 03FE8 150A0001 (01905)         ADD(BR0,H$,TE)          ;... AND MSB WORD
A0B 03FEA 16190981 (01906)         SUBL(BP1,1),JUMPP(01)   ;LOOP IF MORE TO DO
A0C 03FEC 18200031 (01907)         CLEAR(RI)
A0D 03FEE 1A000020 (01908)         NOP(0)
                             (01909) ;
                             (01910) ;OUTPUT PROGRAM:
                             (01911) ;  BW0: RSRB(N)
                             (01912) ;  BW1: LOOP COUNTER (INIT TO NRESID-1)
                             (01913) ;
A0E 03FF0 1C300032 (01914) DCDSBO: SET(RA)              ;START APU
A0F 03FF2 1E42B97A (01915)         LOAD(BW0,RSRB-W$(1))    ;INIT BW0 TO POINT TO OUTPUT BUFFER-W$
A10 33FF4 205000D7 (01916)         LOAD(BW1,NRESID-1)      ;INIT LOOP COUNTER
                             (01917) ;MAIN LOOP:
A11 03FF6 228A0002 (01918) 02:     ADD(BW0,W$,TF)          ;GEN OUTPUT ADR
A12 03FF8 24111101 (01919)         SUBL(BW1,1), JUMPP(02)
A13 03FFA 26200030 (01920)         CLEAR(RO)
A14 03FFC 28000020 (01921)         NOP(0)
        03FFE          (01922)         END
              0000002A (01923) DCDSB$Z = 0L - DCDSBI    ;COMPUTE BUS 1 MODULE SIZE
```

```
                (01924) ;
   03FFE        (01925)        END                        ;END OF FILE
```

```
PAGE   581  [BBNB]<BCA16>BBN16U.MSD.2, 29-Dec-88 15:44:23, Ed: WOLF
              APS3-DECODE()       APS PROGRAM FOR DECODE

                         (01924) ;
      03FFE              (01925)          END                          ;END OF FILE
```

```
ACQTMR:     000A (00010) (00945)
AD$BGN:     040A2 (00454) (00476)
AD$CPTA:    0490A (00563) (00608)
AD$CPTB:    04994 (00564) (00615)
AD$DISC:    04986 (00570) (00596)
AD$LOOP:    00004 (00460) (00472)
AD$RTN:     04982 (00575) (00590) (00597)
AD$STOP:    00015 (00465) (00473)
AD$SZ:      00F2C (00452) (00476)
ADASEND:    01F1F (00437) (00464)
ADAMINT:    0495C (00129) (00172) (00570)
ADB$END:    01FF7 (00430) (00471)
ADBPTR:     04952 (00561) (00579)
ADCNTRL:    00002 (00440) (00403)
ADCTR:      00555 (00171) (00310) (00314)
ADPO:       00542 (00270) (00201) (00571) (00573)
AFDT$ORG:   000E0 (00013) (00135) (00140)
AOM$RRDY:   03CA6 (01037) (01046)
AOM$RTN:    03CA2 (01033) (01042) (01049)
AOMINT:     03C06 (00132) (00165) (01020)
B0:         00001 (00017) (00023) (00024) (01203) (01216) (01220) (01227) (01235) (01240) (01259)
B1:         00002 (00010) (00023) (00024) (00026)
B10:        00003 (00023) (01169) (01242) (01274) (01276) (01282) (01288)
B2:         00004 (00019) (00024) (00025) (00026)
B210:       00007 (00024) (01195)
B3:         00008 (00020) (00025) (00026) (01207) (01266)
B32:        0000C (00025) (01273) (01290) (01293)
B321:       0000E (00026) (01221) (01263)
B4:         00010 (00021) (00027)
B5:         00020 (00022) (00027)
B54:        00030 (00027) (01239)
BITS15:     07FFF (00014) (00342) (00343) (00344) (00345) (00346) (00437) (00438) (00494) (00495)
BNO7:       03DA4 (01302) (01307)
C0:         00002 (00029) (00035) (00037)
C1:         00004 (00030) (00035) (00036) (00037) (00039) (01470)
C10:        00006 (00035) (01445) (01452) (01519)
C1HIST:     09C4C (00197) (00198) (01202)
C2:         00008 (00031) (00036) (00037) (00038) (00039) (01470)
C21:        0000C (00036) (01490) (01515)
C210:       0000E (00037) (01463) (01579)
C2HIST:     09C54 (00198) (00199) (01215)
C3:         00010 (00032) (00030) (00039) (01506) (01576)
C32:        00018 (00038) (01513) (01527) (01529)
C321:       0001C (00039) (01476) (01508)
C3HIST:     09C64 (00199) (00200) (01226)
C4:         00020 (00033) (00040)
C5:         00040 (00034) (00040)
C54:        00060 (00040) (01493)
CLEGO:      00002 (00370) (00372)
CLKRATES:   0F4C6 (00350) (00372)
CLKSET:     00000 (00370)
CL0:        00000 (00084) (00590) (00663) (00816) (01042) (01154) (01295) (01533)
CORPAR:     03F12 (00128) (01434)
```

```
CONTAB:      03E00 (01572) (01603)
CSPU$MOS:    021FC (00042) (00130) (00143)
CTWC1:       09000 (00049) (00050) (00051)
CTWC2:       09010 (00050) (00052)
CTWC3:       09000 (00051)
CTWDG:       099F0 (00059) (00060)
CTWG:        09970 (00058) (00059)
CTWK1:       09830 (00052) (00053)
CTWF2:       09000 (00053) (00054)
CTWK3:       090F0 (00054) (00055)
CTWF4:       09910 (00055) (00056)
CTWK5:       09930 (00056) (00057)
CTWF6:       09950 (00057) (00058)
CTWW:        099F0 (00060) (00061)
D16$INT1:    04022 (00076) (00149)
D16$INT2:    04030 (00077) (00156)
D22$INT1:    040FA (00078) (00163)
D23$INT1:    0411E (00079) (00170)
DA$NGH:      040E2 (00510) (00530)
DA$CPVA:     03CAE (01619) (01052)
DA$CPVB:     03CB0 (01020) (01059)
DA$LOOP:     00001 (00512) (00526)
DA$STOP:     00015 (00510) (00527)
DA$SZ:       0002C (00508) (00530)
DAA$END:     03EA7 (00494) (00517)
DAB$END:     03F7F (00495) (00525)
DADPTR:      03C7C (01020) (01030) (01040)
DACNTRL:     00002 (00497) (00537)
DACTR:       00558 (00164) (00313)
DAPO:        00549 (00200) (01029) (01031)
DCD$00:      00012 (01793)
DCD$01:      00014 (01792) (01796)
DCD$0X:      00011 (01792)
DCD$10:      00017 (01800)
DCD$11:      00019 (01799) (01803)
DCD$1X:      00016 (01791) (01799)
DCD$OUT:     0001A (01794) (01797) (01801) (01804)
DCDSA:       03FA0 (00137) (01036)
DCDSA$Z:     0002A (01020) (01060)
DCDSAI:      03FA0 (01037) (01060)
DCDSAO:      0000E (01030) (01059)
DCDSB:       03FD4 (00142) (01091)
DCDSB$Z:     0002A (01083) (01923)
DCDSBI:      03FD4 (01092) (01923)
DCDSBO:      0000E (01093) (01914)
DCDU:        03F5A (00136) (00141) (01773)
DCDU$SA:     00000 (01770) (01774) (01809)
DCDU$SZ:     0001F (01771) (01809)
DCHIST:      09C40 (00196) (00197) (01176) (01103) (01100)
DVLC1:       09A00 (00061) (00062) (00063) (01479)
DVLC2:       09A10 (00062) (00064) (01480)
DVLC3:       09A00 (00063) (01491)
DVLDG:       09AF0 (00071) (00072) (01449) (01454) (01455)
```

```
 DVLG:       09A70 (00070) (00071) (01440)
 DVLK1:      09A30 (00064) (00065) (01509)
 DVLF2:      09A80 (00065) (00066) (01510)
 DVLK3:      09AF0 (00066) (00067) (01516)
 DVLK4:      09B10 (00067) (00068) (01520)
 DVLF5:      09B30 (00068) (00069) (01530)
 DVLK6:      09B50 (00069) (00070) (01531)
 DVLU:       09BF0 (00072) (00073)
 DVLUF:      09C00 (00073) (01841) (01896)
 EDCORR:     03EC0 (01572)
 ERRPTR:     03BE6 (00036) (00050) (00052)
 ERRSYNC:    03BE7 (00057) (00952) (00986) (00993)
 FDTS:       007E0 (00001) (00125)
 G0:         00004 (00005)
 G1:         00005 (00006) (01130) (01154) (01159) (01295) (01434) (01533)
 G2:         00006 (00007) (00570) (00590) (00640) (00663) (00755) (00016) (01020) (01042)
 G3:         00007 (00008)
 GET1:       03FD6 (01460) (01472) (01482) (01485) (01495) (01502) (01596)
 GET2:       03FCC (01510) (01522) (01524) (01586)
 GET74:      03EA0 (01440) (01444) (01451) (01457) (01462) (01475) (01480) (01498) (01505) (01512)
                   (01526) (01549)
 GHIST:      09C08 (00195) (00196) (01160)
 GT1:        03FD1 (01590) (01597)
 H$:         00001 (00007) (01162) (01164) (01430) (01049) (01050) (01904) (01905)
 H000:       03E02 (01323) (01307)
 H017:       03DF0 (01330) (01300)
 H026:       03DDE (01337) (01351)
 H031:       03DE4 (01332) (01357)
 H045:       03DDC (01320) (01349)
 H052:       03DF4 (01325) (01373)
 H063:       03DE5 (01334) (01350)
 H074:       03DFD (01335) (01302)
 H103:       03DF9 (01326) (01370)
 H114:       03DE3 (01327) (01356)
 H125:       03DF3 (01336) (01372)
 H132:       03DF0 (01333) (01369)
 H146:       03DE2 (01329) (01355)
 H151:       03DDA (01324) (01347)
 H160:       03DFF (01331) (01304)
 H177:       03DEA (01330) (01363)
 HISTO:      09C00 (00194) (00195)
 HPTR:       03DB2 (01305) (01323)
 IDW$CS:     00F10 (00094)
 IDW$DM:     00000 (00095)
 IDW$IC:     00020 (00093)
 IDW$DMHF:   00000 (00091) (00764)
 IDW$RD:     00001 (00090) (00837) (00920) (00955) (00995)
 IDW$RR:     00004 (00096)
 IDW$SI:     00040 (00092)
 IDW$SQ:     00002 (00097)
 ISVT$:      00502 (00100) (00252) (00253) (00254) (00255) (00256) (00257) (00258) (00259) (00270)
                   (00271) (00273) (00274) (00276) (00277) (00279) (00280) (00286) (00287) (00288)
                   (00289) (00293) (00298) (00299) (00300) (00301) (00302) (00303) (00310) (00311)
```

```
                    (00312) (00313) (00319) (00321)
  K1HIST:    09C6C  (00200) (00201) (01234)
  K2HIST:    09CAC  (00201) (00202) (01247)
  K3HIST:    09CCC  (00202) (00203) (01250)
  K4HIST:    09CDC  (00203) (00204) (01272)
  K5HIST:    09CEC  (00204) (00205) (01281)
  K6HIST:    09CFC  (00205) (00206) (01287)
  LASTERR:   03DE0  (00050) (00953) (00983) (00987) (00992)
  LOSETHR:   00004  (00101) (00951) (00985)
  MALNGTH:   0020A  (00180) (00342) (00343) (00344) (00345) (00346) (00798) (00871) (00887) (00917)
  ML18:      00034  (00189) (00606) (00688) (00689) (00690) (00691) (00697) (00699) (00780) (00781)
                    (00782)
  MODSBGN:   04002  (00360) (00423)
  MODSINIT:  00006  (00373) (00376)
  MODSLOOP:  0000A  (00380) (00382) (00390) (00393) (00396) (00399) (00402) (00405) (00412) (00414)
                    (00417) (00419)
  MODSRA:    00015  (00376) (00389) (00403)
  MODSRB:    0001C  (00387) (00387) (00391) (00395)
  MODSRC:    00022  (00388) (00388) (00397) (00401)
  MODSRCVR:  00010  (00380) (00386)
  MODSSZ:    0006F  (00366) (00423)
  MODSTA:    00028  (00411)
  MODSTB:    00031  (00409) (00416)
  MODSXPTR:  00028  (00381) (00409)
  NEWMAX:    03C22  (00926)
  NOCNRR:    03EC4  (01571) (01576)
  NOMAX:     03C28  (00925) (00932)
  NRESID:    000D8  (00190) (00878) (00879) (01142) (01162) (01164) (01429) (01430) (01438) (01847)
                    (01861) (01902) (01916)
  ODWSRS:    00004  (00107) (00114)
  ODWSSD:    00001  (00109)
  ODWSSR:    00002  (00108)
  ODWSTR:    00000  (00106) (00114)
  OLDSYNC:   03DE9  (00859) (00956) (00980) (00996)
  OUT2:      03DA0  (01277) (01283) (01289) (01312)
  OUT7:      03D9C  (01172) (01179) (01191) (01198) (01210) (01222) (01243) (01254) (01267) (01275)
                    (01294) (01302)
  P0:        03E03  (01388)
  P1:        03DE0  (01361)
  P2:        03DDF  (01352)
  P3:        03DF1  (01370)
  P4:        03E01  (01386)
  P5:        03DD0  (01348)
  P6:        03DE6  (01359)
  P7:        03DE8  (01364)
  PROPAR:    03CE4  (00127) (01159)
  PROWES:    03CCA  (00126) (01130)
  PTR2:      03D02  (01314) (01341)
  Q0:        03E04  (01341) (01389)
  Q1:        03DE9  (01342) (01362)
  Q2:        03DE0  (01343) (01353)
  Q3:        03DF2  (01344) (01371)
  RANDSL:    000E0  (00111) (00849) (00852)
```

```
RANDSUB:     Ø1C77 (ØØ112) (ØØ847)
RBQFD:       ØØ551 (ØØ3Ø3) (ØØ785) (ØØ791) (ØØ795) (ØØ95Ø) (ØØ977)
RBTA:        ØØ39C (ØØ226) (Ø1426) (Ø1846)
RBTB:        ØØ5A6 (ØØ227) (Ø1427) (Ø1961)
RBTBPTR:     Ø3E8C (ØØ7ØØ) (ØØ8Ø2) (Ø1427) (Ø1436)
RBTFA:       ØØ53B (ØØ256) (ØØ262) (ØØ739)
RBTFB:       ØØ53C (ØØ257) (ØØ74Ø)
RBTFPTR:     Ø3B5Ø (ØØ74Ø) (ØØ777) (ØØ8Ø1)
RBTPD:       ØØ547 (ØØ277) (ØØ776) (ØØ8Ø3) (ØØ8ØØ)
RDADA:       ØBDDØ (ØØ233) (ØØ494) (ØØ513) (Ø1Ø53) (Ø1Ø54) (Ø1Ø55)
RDADB:       ØBEAØ (ØØ234) (ØØ495) (ØØ521) (Ø1Ø6Ø) (Ø1Ø61) (Ø1Ø62)
RERRS:       ØØ57F (ØØ321) (ØØ766)
RFCTR:       ØØ54F (ØØ299) (ØØØØ4)
RLCTR:       ØØ55Ø (ØØ3ØØ) (Ø1ØØ1)
RMA$END:     Ø2F87 (ØØ342) (ØØ39Ø)
RMB$END:     Ø3191 (ØØ343) (ØØ396)
RMC$END:     Ø339B (ØØ344) (ØØ4Ø2)
RMCTR:       ØØ557 (ØØ157) (ØØ312)
RMDMA:       ØAD7E (ØØ223) (ØØ342) (ØØ377) (ØØ4Ø4) (ØØ743)
RMDMB:       ØAF8Ø (ØØ224) (ØØ343) (ØØ392) (ØØ744)
RMDMC:       ØB192 (ØØ225) (ØØ344) (ØØ398) (ØØ742) (ØØ745)
RMFDC:       ØØ54C (ØØ28Ø) (ØØ822)
RMI$BMD1:    Ø3BA8 (ØØ789) (ØØ794)
RMI$BMD2:    Ø3BAA (ØØ793) (ØØ8ØØ)
RMI$DISC:    Ø3BC6 (ØØ77Ø) (ØØ822)
RMI$LOST:    Ø3BBA (ØØ775) (ØØ812)
RMI$RTN:     Ø3BC2 (ØØ762) (ØØ8Ø5) (ØØ8Ø9) (ØØ816) (ØØ823)
RMI$SIN:     Ø3BCA (ØØ768) (ØØ834)
RMI$SINI:    Ø3BB6 (ØØ771) (ØØ8ØØ)
RMI$SRCH:    Ø3BC8 (ØØ772) (ØØ815)
RMI$STF:     Ø3BAC (ØØ796) (ØØ8Ø1)
RMODENIN:    Ø3B62 (ØØ131) (ØØ158) (ØØ754)
RMPD:        ØØ546 (ØØ276) (ØØ756) (ØØ76Ø)
RMPTR:       Ø3B6Ø (ØØ745) (ØØ763) (ØØ784) (ØØ797) (ØØ812) (ØØ814)
RNCOR:       ØØ57D (ØØ319) (ØØ323) (Ø157Ø)
RONNK:       ØØ551 (ØØ3Ø1) (ØØ58Ø) (ØØ765)
RSIBPTR:     Ø3CØØ (Ø1Ø23) (Ø1Ø35)
RSIFPTR:     Ø3C84 (Ø1Ø25) (Ø1Ø36) (Ø1Ø39)
RSNFA:       ØØ53D (ØØ258) (Ø1Ø24)
RSNFB:       ØØ53E (ØØ259) (Ø1Ø25)
RSNKA:       ØØB4Ø (ØØ23Ø) (Ø1Ø22)
RSNKB:       ØØC2Ø (ØØ231) (Ø1Ø23)
RSNKC:       ØØCFØ (ØØ232) (Ø1Ø47)
RSNNR:       ØØ54D (ØØ289) (Ø1Ø46)
RSNPD:       ØØ54Ø (ØØ279) (Ø1Ø34) (Ø1Ø41)
RSRA:        ØØ7BØ (ØØ228) (ØØ878) (Ø1429) (Ø1869)
RSRB:        ØB97C (ØØ229) (ØØ879) (Ø143Ø) (Ø1915)
RSRBPTR:     Ø3E1Ø (Ø143Ø) (Ø1439)
RSSPF:       Ø7BEC (ØØ238) (ØØ874) (ØØ921) (ØØ938)
RSSSS:       Ø7DF6 (ØØ239) (ØØ889) (ØØ89Ø) (ØØ891) (ØØ923) (ØØ924) (ØØ926) (ØØ937)
RSTNC:       ØØ552 (ØØ3Ø2) (ØØ769) (ØØ774) (ØØ877) (ØØ949) (ØØ96Ø) (Ø1ØØØ)
RUN:         ØØ53F (ØØ293) (ØØ295) (ØØ574) (ØØ652) (ØØ761) (Ø1Ø32)
RUNMOD:      ØØØØ1 (ØØ371)
```

```
PAGE   56: (BBNU)<DCA16>BBN16U.NSO.2, 29-Dec-80 15:44:23, Ed: WOLF
           APS3-DECODE()       APS PROGRAM FOR DECODE

  SALNGTH:    000D0 (00186) (00187) (00190) (00437) (00438) (00494) (00495)
  SET:        00020 (00083) (00570) (00640) (00755) (01020) (01138) (01159) (01434)
  SISNUM:     03C00 (00085) (00088)
  SISLOOP:    03BEB (00073) (00076)
  SL6:        00024 (00107) (00608) (00610) (00611) (00615) (00617) (00618) (01052) (01054) (01055)
                    (01059) (01061) (01062)
  SSSALT:     03C1C (00923)
  SSSLOOP:    03C16 (00920) (00941)
  SSSLPTST:   03C2E (00929) (00932) (00934) (00938)
  SSSNO:      03C4E (00946) (00948) (00960)
  SSSNOALT:   03C2C (00922) (00937)
  SUSFLF:     03C62 (00984) (00987)
  SUSFRR:     03C66 (00982) (00991)
  SUSLOSE:    03C74 (00994) (01000)
  SUSVALID:   03C6E (00980) (00995)
  SYNCINIT:   03BEA (00000) (00013) (00071)
  SYNCSRCH:   03C12 (00015) (00917)
  SYNCUPDA:   03C52 (00773) (00977)
  SYSSFLGS:   1FFCE (00113) (00570) (00590) (00640) (00663) (00755) (00816) (01020) (01042) (01130)
                    (01154) (01159) (01295) (01434) (01533)
  TADRA:      09D40 (00210) (00437) (00460) (00560)
  TADRB:      09E20 (00211) (00438) (00467) (00561)
  TADRC:      09EFA (00212) (00502)
  TADRC:      0054A (00286) (00305) (00596)
  TBTA:       0A34C (00217) (01134)
  TBTB:       0A556 (00218) (01135)
  TBTBPTR:    03CC0 (00657) (01135) (01141) (01163)
  TBTC:       01760 (00219) (00671)
  TBTFA:      00536 (00254) (00640)
  TBTFB:      00537 (00255) (00641)
  TBTFPTR:    03A02 (00641) (00655) (00660)
  TBTPU:      00544 (00273) (00654) (00662)
  TFCTR:      F054E (00290) (00509)
  TMASEND:    02B73 (00345) (00412)
  TMBSEND:    0207D (00346) (00417)
  TMBITS:     0000C (00114) (00243) (00245) (01146) (01149) (01204) (01217) (01228) (01236) (01249)
                    (01260) (01347) (01348) (01349) (01350) (01351) (01352) (01353) (01354) (01355)
                    (01356) (01357) (01358) (01359) (01360) (01361) (01362) (01363) (01364) (01365)
                    (01366) (01367) (01368) (01369) (01370) (01371) (01372) (01373) (01374) (01375)
                    (01376) (01377) (01378) (01379) (01380) (01381) (01382) (01383) (01384) (01385)
                    (01386) (01387) (01388) (01389) (01390) (01391) (01392) (01393)
  TMCTR:      00556 (00150) (00311)
  TMDMA:      0A96A (00220) (00242) (00345) (00379) (00418) (00685) (00687) (00688) (00689) (00690)
                    (00691)
  TMDMB:      0AB74 (00221) (00244) (00346) (00413) (00696) (00698) (00699) (00700) (00701) (00702)
  TMFFC:      0054B (00287) (00670)
  TMISCPVA:   03A32 (00643) (00684)
  TMISCPVB:   03B44 (00644) (00695)
  TMISFAKE:   03A2A (00656) (00670)
  TMISRTN:    03A26 (00653) (00663) (00673)
  TMODENIN:   03B08 (00130) (00151) (00647)
  TMPU:       00545 (00274) (00649) (00651)
  TMPTR:      03A06 (00644) (00659) (00672)
```

```
PAGE   57: [BBND]<DCA16>BBN16U.MSO.2, 29-Dec-80 15:44:23, Ed: WOLF
             APS3-DECODR()       APS PROGRAM FOR DECODR

  TOPHIST:     09D0C (00206)
  TSNRPTR:     03CC4 (01132) (01140) (01161)
  TSNKA:       0A100 (00215) (01131)
  TSNKB:       0A266 (00216) (01132)
  TSRA:        09FD0 (00213) (00609) (00610) (00611)
  TSRB:        0A0A0 (00214) (00616) (00617) (00618)
  TSRRPTR:     04956 (00564) (00584)
  TSNFA:       00534 (00252) (00260) (00566)
  TSRFB:       00535 (00253) (00567)
  TSRFPTR:     0495A (00567) (00577) (00586)
  TSNPD:       00543 (00271) (00576) (00580)
  TWON9:       03FCA (01840) (01870) (01895)
  WS:          00E02 (00008) (00050) (00052) (00053) (00054) (00055) (00056) (00057) (00058) (00059)
                     (00060) (00061) (00062) (00064) (00065) (00066) (00067) (00068) (00069) (00070)
                     (00071) (00072) (00073) (00125) (00135) (00140) (00878) (00879) (00884) (00886)
                     (00890) (00891) (01429) (01430) (01842) (01843) (01844) (01860) (01863) (01897)
                     (01898) (01899) (01915) (01918)

LINES WITH ERRORS:     0  (MAP VERSION 000101.10)   E-     0
```

```
C  <DCA16>BBN16C.FOR.1    Mon 10-Nov-88 6:32PM          Page 1

C <BBNO><DCA16>BBN16C.FOR.1, 10-Nov-88 10:32:22, Ed: KFIELD
        SUBROUTINE BBN16C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                          BBN16C
C       BBN16 MAP-300 BUFFER & SCALAR CONFIGURATIONS
C
C       CALLED BY BBN16R
C
C KDFIELD 7/88
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C MAP-RELATED VARIABLES:
C
        REAL MWSZ,FWSZ,DWSZ
        INTEGER RL,CMPLX,FID,CNTG,LNG,SHRT
        INTEGER PRCSR,AP
C
C
C <BBNO><MAP>BBN16-COMMON.FOR.12, 31-Oct-88 23:17:03, Ed: WOLF
C
C
C MAP BUFFER NAMES:
C
        INTEGER DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3
        INTEGER DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU
        INTEGER DVLUF
        INTEGER CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3
        INTEGER CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU
        INTEGER HISTO
        INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, TONES
        INTEGER TSP0, TSP1, TSP,  THAN0,TWSP, TWSPZ
        INTEGER TCUST,TXSP, TXSPH,TXSPI,TPSP, TPSPH
        INTEGER TPSPI,TRP,  TC,   TWORK,TIC,  TCH
        INTEGER TE10, TE11, TE1,  THAN1,TWE1, TRS
        INTEGER TLHU, TRSP, TKE,  TKH,  TIK,  TFILT
        INTEGER TAN,  TANR, TE2,  TFAC, TANS, TGFAC
        INTEGER TIDG, TUO,  TU1,  TVH,  TRHB, TRH
        INTEGER TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB
        INTEGER TBTC, THDHA,THDHB,THPP
        INTEGER RHDHA,RHDHB,RHDHC,RBTA, RBTB, RSRA
        INTEGER RSRB, RUHA, RUHB, RDGH, RCH,  RKH
        INTEGER RWH,  RVHH, RVH,  RRHB, RRH,  RRHDA
        INTEGER HRHDH,HSNKA,RSNKB,HSNKC,HDABA,HDABB
C
C
C MAP REAL SCALAR NAMES:
```

```
C  <DCA16>BBN16C.FOR.1    Mon 10-Nov-88 6:32PM          Page 1:1

C
        INTEGER TALPH,TPHST,TOCH, TFSZI,THH,  TIH
        INTEGER T14,  TKTHR,TIG,  TC2P, T72R, T3R
        INTEGER RGH,  RHH,  RDCF0,RDCF1,RDCF2,RDCF3
        INTEGER RDHY0,RDHY1,RDHY2,RDHY3
C
C
C MAP INTEGER SCALAR NAMES:
C
        INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG
        INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RUN
        INTEGER ADPO, TSRPO,TBTPO,THPO, RHPO, RBTPO
        INTEGER RSNPO,DAPO, TADDC,THFFC,RHFDC,RSNNR
        INTEGER TFCTR,RFCTR,RLCTR,RONHK,RSYNC,RBUFO
        INTEGER ADCTR,THCTR,RHCTR,DACTR,TADPK
        INTEGER RNCOR,RERRS
C
C
C COMMON BLOCK - BUFFER AND SCALAR NAMES:
C
        COMMON /BSIDS/
     $  DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3,
     $  DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU,
     $  DVLUF,
     $  CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3,
     $  CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU,
     $  HISTO,
     $  TADBA,TADBB,TADBC,TSRA, TSRB, TONES,
     $  TSP0, TSP1, TSP,  THAN0,TWSP, TWSPZ,
     $  TCUST,TXSP, TXSPH,TXSPI,TPSP, TPSPH,
     $  TPSPI,TRP,  TC,   TWORK,TIC,  TCH,
     $  TE10, TE11, TE1,  THAN1,TWE1, TRS,
     $  TLHU, TRSP, TKE,  TKH,  TIK,  TFILT,
     $  TAN,  TANR, TE2,  TFAC, TANS, TGFAC,
     $  TIDG, TUO,  TU1,  TVH,  TRHB, TRH,
     $  TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB,
     $  TBTC, THDHA,THDHB,THPP,
     $  RHDHA,RHDHB,RHDHC,RBTA, RBTB, RSRA,
     $  RSRB, RUHA, RUHB, RDGH, RCH,  RKH,
     $  RWH,  RVHH, RVH,  RRHB, RRH,  RRHDA,
     $  HRHDH,RSNKA,HSNKB,HSNKC,HDABA,RDABB,
     $  TALPH,TPHST,TOCH, TFSZI,THH,  TIH,
     $  T14,  TKTHR,TIG,  TC2P, T72R, T3R,
     $  RGH,  RHH,  RDCF0,RDCF1,RDCF2,RDCF3,
     $  RDHY0,RDHY1,RDHY2,RDHY3,
     $  TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG,
     $  RBTFA,RBTFB,RSNFA,RSNFB,RUN,
     $  ADPO, TSRPO,TBTPO,THPO, RHPO, RBTPO,
     $  RSNPO,DAPO, TADDC,THFFC,RHFDC,RSNNR,
     $  TFCTR,RFCTR,RLCTR,RONHK,RSYNC,RBUFO,
     $  ADCTR,THCTR,RHCTR,DACTR,TADPK,
     $  RNCOR,RERRS
```

AD-A096 092 BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA F/G 17/2.1
DESIGN AND REAL-TIME IMPLEMENTATION OF A ROBUST APC CODER FOR S--ETC(U)
DEC 80 J J WOLF, K D FIELD, W H RUSSELL DCA100-79-C-0037
UNCLASSIFIED BBN-4565-VOL-2 NL

5 OF 5
AD A
096092

END
DATE
FILMED
4-81
DTIC

```
C  <UCA16>BBN16C.FOR.1   Mon 18-Nov-85 6:32PM          Page 1:2

C
C
C COMMON BLOCK - BUFFER ADDRESSES AND SIZES
C        NEEDED FOR INIT OF NON-SNAP BUFFERS
C        AND FOR FILE-TO-FILE MODE OF OPERATION
C
         COMMON /MAS/
     )   AHISTO,SHISTO,
     )   ATADBA,STADBA,
     )   ATADBB,STADBB,
     )   ATADBC,STADBC,
     )   ATBTC, STBTC,
     )   ATNDMA,STNDMA,
     )   ATNDMB,STNDMB,
     )   ARNDMA,SRNDMA,
     )   ARNDMB,SRNDMB,
     )   ARNDMC,SRNDMC,
     )   ARSNKC,SRSNKC,
     )   ARDABA,SRDABA,
     )   ARDABB,SRDABB
C
CCCCCCC END OF BBN16-COMMON CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C INITIALIZE FORTRAN CONSTANTS:
C
         DATA PNCSH/1/,AP/1/
         DATA NL/8/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/8/,SHRT/1/
         DATA HWSZ/1./,FWSZ/2./,DWSZ/4./
         DATA IBUS1/64/,IBUS2/128/,IBUS3/192/
C
C
C
C DEFINE FUNCTION EVEN(X)
C RETURNS SMALLEST EVEN INTEGER .GE. X
C (ASSUMES X .EQ. AINT(X) )
C
         EVEN(X) = X + (X - (2.*AINT(X/2.)))
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CONFIGURE MAP-300 BUFFERS:
C
C THIS MODULE CONFIGURES THE SYSTEM MAP BUFFERS.
C ALL MAP BUFFERID'S (BID'S) ARE SYMBOLICALLY
C NAMED, WITH NAMES STARTING WITH "T" FOR
C TRANSMITTER BUFFERS, AND NAMES STARTING WITH
C "R" FOR RECEIVER BUFFERS.
C AN "A" FOLLOWED BY A BUFFER NAME SPECIFIES
```

```
C  <UCA16>BBN16C.FOR.1   Mon 18-Nov-85 6:32PM          Page 1:3

C THE HOST VARIABLE CONTAINING THE ADDRESS OF
C THAT BUFFER.  SIMILARLY, AN "S" FOLLOWED BY A
C BUFFER NAME SPECIFIES THE HOST VARIABLE
C CONTAINING THE SIZE OF THAT BUFFER.
C
C A BID OF 0 INDICATES A NON-SNAP BUFFER.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DEFINE SYMBOLIC BID'S:
C
C
C DECODED VALUE TABLE OF QUANTIZED PITCH TAPS C1 & C3 (3 BITS)
         DVLC1 = 0
         DVLC3 = DVLC1
C
C DECODED VALUE TABLE OF QUANTIZED PITCH TAP C2 (4 BITS)
         DVLC2 = 0
C
C (DVLK1-6 MUST BE PHYSICALLY CONTIGUOUS IN MEMORY)
C DECODED VALUE TABLE OF QUANTIZED REFLECTION COEF K1 (6 BITS)
         DVLK1 = 0
C
C DECODED VALUE TABLE OF QUANTIZED REFLECTION COEF K2 (5 BITS)
         DVLK2 = 0
C
C DECODED VALUE TABLE OF QUANTIZED REFLECTION COEF K3 (4 BITS)
         DVLK3 = 0
C
C DECODED VALUE TABLE OF QUANTIZED REFLECTION COEF K4 (4 BITS)
         DVLK4 = 0
C
C DECODED VALUE TABLE OF QUANTIZED REFLECTION COEF K5 (4 BITS)
         DVLK5 = 0
C
C DECODED VALUE TABLE OF QUANTIZED REFLECTION COEF K6 (4 BITS)
         DVLK6 = 0
C
C DECODED VALUE TABLE OF SQRT(QUANTIZED ENERGY)
         DVLG  = 0
C
C DECODED VALUE TABLE OF SQRT(QUANTIZED DELTA GAINS)
         DVLDG = 0
C
C DECODED VALUE TABLE OF QUANTIZED RESIDUAL SAMPLES
         DVLU  = 0
C
C DECODED VALUE TABLE OF QUANTIZED RESID SAMPLES (FOLDED BINARY ORDER)
         DVLUP = 0
C
```

```
C  <DCA16>BBN16C.FOR.1   Mon 18-Nov-80 6:32PM         Page 1:4

C CODING TABLE OF PITCH TAP THRESHOLDS C1 & C3
        CTHC1 = 0
        CTHC3 = CTHC1
C
C CODING TABLE OF PITCH TAP THRESHOLDS C2
        CTHC2 = 0
C
C (CTHK1-6 MUST BE PHYSICALLY CONTIGUOUS IN MEMORY)
C CODING TABLE OF K1 THRESHOLDS
        CTHK1 = 0
C
C CODING TABLE OF K2 THRESHOLDS
        CTHK2 = 0
C
C CODING TABLE OF K3 THRESHOLDS
        CTHK3 = 0
C
C CODING TABLE OF K4 THRESHOLDS
        CTHK4 = 0
C
C CODING TABLE OF K5 THRESHOLDS
        CTHK5 = 0
C
C CODING TABLE OF K6 THRESHOLDS
        CTHK6 = 0
C
C CODING TABLE OF ENERGY THRESHOLDS
        CTHG  = 0
C
C CODING TABLE OF DELTA GAIN THRESHOLDS
        CTHDG = 0
C
C CODING TABLE OF RESIDUAL SAMPLE THRESHOLDS (FOLDED BINARY)
        CTHU  = 0
C
C HISTOGRAM BUFFER FOR K'S (DEBUGGING AND SYSTEM MEASUREMENT USE)
        HISTO = 0
C
C A/D INPUT FROM ADAM
        TADHA = 0
C
C A/D INPUT FROM ADAM
        TADBB = 0
C
C A/D "ON-HOOK" TONE DATA
        TADBC = 0
C
C INPUT SPEECH TO ANLZA
        TSRA  = 1
C
C INPUT SPEECH TO ANLZB
        TSRB  = 2
```

```
C  <DCA16>BBN16C.FOR.1   Mon 18-Nov-80 6:32PM         Page 1:5

C
C UNITY VECTOR
      TONES = 3
C
C PREEMPHASIZED SPEECH (CURRENT FRAME)
      TSP0  = 4
C
C PREEMPHASIZED SPEECH (LAST FRAME AND CURRENT FRAME)
      TSP1  = 5
C
C PREEMPHASIZED SPEECH (LAST 49 OF LAST FRAME, ALL OF CURRENT FRAME)
      TSP   = 6
C
C HANNING WINDOW COEFFICIENTS FOR 265 SAMPLE WINDOW
      THANW = 7
C
C WINDOWED SPEECH
      TWSP  = 8
C
C WINDOWED SPEECH WITH APPENDED ZEROS
      TWSPZ = 9
C
C COSINE TABLE FOR FFT AND IFFT
      TCOST = 10
C
C FREQUENCY DOMAIN REPRESENTATION OF TWSPZ
C (THIS BUFFER CONTAINS 256 COMPLEX POINTS; A 257TH COMPLEX
C  POINT WILL BE ASSUMED PRESENT DIRECTLY FOLLOWING IT)
      TXSP  = 11
C
C REAL PART OF TXSP, INCLUDING 257TH POINT
      TXSPR = 12
C
C IMAGINARY PART OF TXSP, INCLUDING 257TH POINT
      TXSPI = 13
C
C COMPLEX BUFFER OF TPSPR,TPSPI
C (THIS BUFFER CONTAINS 256 COMPLEX POINTS; A 257TH COMPLEX
C  POINT WILL BE ASSUMED PRESENT DIRECTLY FOLLOWING IT)
      TPSP  = 14
C
C POWER SPECTRUM OF TXSP, COMPRISING REAL PART OF TPSP,
C  INCLUDING 257TH POINT.
      TPSPR = 15
C
C ZEROS, COMPRISING IMAGINARY PART OF TPSP, INCLUDING 257TH POINT
      TPSPI = 16
C
C AUTOCORRELATION COEFFS FOR PITCH DETERMINATION
      TRP   = 17
C
```

```
C  <DCA16>BBN16C.FOR.1    Mon 10-Nov-80 6:32PM         Page 1:6

C PITCH PREDICTOR COEFFS (TAPS)
        TC     = 18
C
C AUTOCORRELATION COEFFS:  -TRP(PITCH+N)); N=-1,0,+1
        TRPP  = 19
C
C WORK AREA FOR MCOL
        TWORK = 20
C
C CODED PITCH PREDICTOR COEFFS
        TIC    = 21
C
C QUANTIZED PITCH PREDICTOR COEFFS
        TCN    = 22
C
C FIRST RESIDUAL (CURRENT FRAME)
        TE10  = 23
C
C FIRST RESIDUAL (LAST 6 SAMPLES OF CURRENT FRAME)
        TE11  = 24
C
C FIRST RESIDUAL (LAST 6 SAMPLES OF LAST FRAME, PLUS CURRENT FRAME)
        TE1    = 25
C
C HANNING WINDOW COEFFS FOR 216 SAMPLE WINDOW
        THAN1 = 26
C
C WINDOWED FIRST RESIDUAL
        TWE1  = 27
C
C AUTOCORRELATION COEFFS FOR SPECTRAL ANALYSIS
        TRS    = 28
C
C LAMBDA * MU(N) FOR NFC MODULE
        TLMU  = 29
C
C NFC'D AUTOCORRELATION COEFFS
        TRSP  = 30
C
C REFLECTION COEFFS & ERROR TERMS FROM MWLQ6
        TKE    = 31
C
C QUANTIZED REFLECTION COEFFS
        TKN    = 32
C
C CODED REFLECTION COEFFS
        TIK    = 33
C
C PREDICTOR COEFFS A(1)-A(6) (ALSO START OF FILTER COEFF BLOCK)
        TAN    = 34
        TFILT = TAN
```

```
C  <DCA16>BBN16C.FOR.1    Mon 10-Nov-80 6:32PM         Page 1:7

C
C PREDICTOR COEFFS (REVERSE BUFFER), WITH A(0)=1 ADDED
        TANR  = 35
C
C SECOND RESIDUAL
        TE2    = 36
C
C FACTORS FOR NOISE SHAPING ANALYSIS
        TFAC  = 37
C
C NOISE SHAPING FILTER COEFFS
        TANS  = 38
C
C QUANTIZER SCALE FACTORS, ALTERNATING WITH THEIR RECIPROCALS
        TGFAC = 39
C
C CODED DELTA GAINS
        TIDG  = 40
C
C UNQUANTIZED, NORMALIZED RESIDUAL (FOR DEBUGGING ONLY)
        TUO    = 41
C
C HISTORY FOR NOISE-SHAPING FILTER
        TQL    = 42
C
C HISTORY FOR SPECTRAL FILTER
        TVN    = 43
C
C APC PITCH FILTER HISTORY (CURRENT FRAME)
        TRNW  = 44
C
C APC PITCH FILTER HISTORY (LAST FRAME PLUS CURRENT FRAME)
        TRN    = 45
C
C CODED RESIDUAL SAMPLES FROM ANLZA
C (ALSO, START OF CODED TRANSMISSION DATA BUFFER FROM ANLZA,
C  WITH FORMAT TIUA,TIG,TIDG,TIN,TIC,TIK)
        TIUA  = 46
        TSNKA = TIUA
C
C CODED RESIDUAL SAMPLES FROM ANLZB
C (ALSO, START OF CODED TRANSMISSION DATA BUFFER FROM ANLZB,
C  WITH FORMAT TIUB,TIG,TIDG,TIN,TIC,TIK)
        TIUB  = 47
        TSNKB = TIUB
C
C TBITS A BUFFER
        TBTA  = 0
C
C TBITS B BUFFER
        TBTB  = 0
```

```
C
C TBITS C BUFFER
        TBTC  = 0
C
C TMODEM A BUFFER
        TMDMA = 0
C
C TMODEM B BUFFER
        TMDMB = 0
C
C RMODEM A BUFFER
        RMDMA = 0
C
C RMODEM B BUFFER
        RMDMB = 0
C
C RMODEM C BUFFER
        RMDMC = 0
C
C RBITS A BUFFER
        RBTA  = 0
C
C RBITS B BUFFER
        RBTB  = 0
C
C RECEIVED DECODED RESIDUAL SAMPLES FOR SYNZA.
C ALSO, START OF RECEIVED DECODED TRANSMISSION DATA FOR SYNZA:
C  RUNA,RGN,RDGN,RNN,RCN,RKN
        RSRA  = 48
        RUNA  = RSRA
C
C RECEIVED DECODED RESIDUAL SAMPLES FOR SYNZB.
C ALSO, START OF RECEIVED DECODED TRANSMISSION DATA FOR SYNZB:
C  RUNB,RGN,RDGN,RNN,RCN,RKN
        RSRB  = 49
        RUNB  = RSRB
C
C RECEIVED DECODED SQRT(DELTA GAINS)
        RDGN  = 50
C
C RECEIVED DECODED PITCH PREDICTOR COEFFS
        RCN   = 51
C
C RECEIVED DECODED REFLECTION COEFFS
        RKN   = 52
C
C SCALED RESIDUAL SAMPLES
        RUN   = 53
C
C SPECTRAL SYNTHESIS FILTER MEMORY
        RVNM  = 54
```



```
C
C OUTPUT OF SPECTRAL SYNTHESIS FILTER
        RVN   = 55
C
C OUTPUT OF PITCH SYNTHESIS FILTER (CURRENT FRAME)
        RRN0  = 56
C
C OUTPUT OF PITCH SYNTHESIS FILTER (LAST FRAME & CURRENT FRAME)
        RRN   = 57
C
C DEEMPHASIZED SYNTHETIC SPEECH FROM SYNZA
        RRNDA = 58
        RSNKA = RRNDA
C
C DEEMPHASIZED SYNTHETIC SPEECH FROM SYNZB
        RRNDB = 59
        RSNKB = RRNDB
C
C SYNTHETIC 'SILENCE' DATA
        RSNKC = 0
C
C D/A OUTPUT TO AOM
        RDADA = 0
C
C D/A OUTPUT TO AOM
        RDADB = 0
C
C
C
C
C DEFINE BUFFER SIZES (NUMBER OF SAMPLES PER BUFFER):
C
        SDVLC1 = 8.
        SDVLC3 = SDVLC1
        SDVLC2 = 16.
        SDVLK1 = 64.
        SDVLK2 = 32.
        SDVLK3 = 16.
        SDVLK4 = 16.
        SDVLK5 = 16.
        SDVLK6 = 16.
        SDVLG  = 64.
        SDVLDG = 4.
        SDVLU  = 4.
        SDVLUP = 4.
        SCTHC1 = 8.
        SCTHC3 = SCTHC1
        SCTHC2 = 16.
        SCTHK1 = 64.
        SCTHA2 = 32.
        SCTHK3 = 16.
```

```
      SCTWK4 = 16.
      SCTWK5 = 16.
      SCTWK6 = 16.
      SCTWG  = 64.
      SCTWDG = 4.
      SCTWW  = 4.
      SWISTO = 108.
      STADBA = 216.
      STADBB = 216.
      STADBC = 216.
      STSRA  = 216.
      STSRB  = 216.
      STUNKS = 216.
      STSPO  = 216.
      STSP1  = 432.
      STSP   = 265.
      STWAN8 = 265.
      STWSP  = 265.
      STWSP2 = 512.
      STCOST = 256.
      STKSP  = 256.
      STKSPR = 257.
      STKSPI = 257.
      STPSP  = 256.
      STPSPR = 257.
      STPSPI = 257.
      STRP   = 512.
      STC    = 3.
      STRPP  = 3.
      STWUNK = 6.
      STIC   = 3.
      STCW   = 3.
      STE10  = 216.
      STE11  = 6.
      STE1   = 222.
      STWAN1 = 216.
      STWE1  = 216.
      STWS   = 7.
      STLWW  = 3.
      STRSP  = 7.
      STKE   = 12.
      STKW   = 6.
      STIE   = 6.
      STAW   = 6.
      STFILT = 15.
      STAWW  = 7.
      STE2   = 216.
      STFAC  = 6.
      STANS  = 6.
      STGFAC = 6.
      STIDG  = 3.
```



```
      STUO   = 216.
      STW1   = 6.
      STVH   = 6.
      STRH0  = 216.
      STRH   = 432.
      STIUA  = 230.
      STSNKA = STIUA
      STIUB  = 230.
      STSNKB = STIUB
      STBTA  = 522.
      STBTB  = 522.
      STBTC  = 522.
      STMDMA = 522.
      STMDMB = 522.
      SRMDMA = 522.
      SRMDMB = 522.
      SRMDMC = 522.
      SRBTA  = 522.
      SRBTB  = 522.
      SRSRA  = 230.
      SRUNA  = SRSRA
      SRSRB  = 230.
      SRUNB  = SRSRB
      SRDGW  = 3.
      SRCW   = 3.
      SRKW   = 8.
      SRWH   = 216.
      SRVHM  = 8.
      SRVW   = 216.
      SRWW0  = 216.
      SRRW   = 432.
      SRRWDA = 216.
      SRSNKA = SRRWDA
      SRRWDB = 216.
      SRSNKB = SRRWDB
      SRSNKC = 216.
      SRDABA = 216.
      SRDABB = 216.
C
C
C
C
C DEFINE BUFFER ADDRESSES (BY BUS):
C
C  BUS1 BUFFERS (START AT $9000 = 36912.):
```

```
C
      BASE1  = 38912.
      ACTWC1 = BASE1
      ACTWC3 = ACTWC1
      ACTWC2 = ACTWC1 + FWSZ*SCTWC1
      ACTWK1 = ACTWC2 + FWSZ*SCTWC2
      ACTWK2 = ACTWK1 + FWSZ*SCTWK1
      ACTWK3 = ACTWK2 + FWSZ*SCTWK2
      ACTWK4 = ACTWK3 + FWSZ*SCTWK3
      ACTWK5 = ACTWK4 + FWSZ*SCTWK4
      ACTWK6 = ACTWK5 + FWSZ*SCTWK5
      ACTWG  = ACTWK6 + FWSZ*SCTWK6
      ACTWDG = ACTWG  + FWSZ*SCTWG
      ACTWU  = ACTWDG + FWSZ*SCTWDG
      ADVLC1 = ACTWU  + FWSZ*SCTWU
      ADVLC3 = ADVLC1
      ADVLC2 = ADVLC1 + FWSZ*SDVLC1
      ADVLK1 = ADVLC2 + FWSZ*SDVLC2
      ADVLK2 = ADVLK1 + FWSZ*SDVLK1
      ADVLK3 = ADVLK2 + FWSZ*SDVLK2
      ADVLK4 = ADVLK3 + FWSZ*SDVLK3
      ADVLK5 = ADVLK4 + FWSZ*SDVLK4
      ADVLK6 = ADVLK5 + FWSZ*SDVLK5
      ADVLG  = ADVLK6 + FWSZ*SDVLK6
      ADVLDG = ADVLG  + FWSZ*SDVLG
      ADVLU  = ADVLDG + FWSZ*SDVLDG
      ADVLUF = ADVLU  + FWSZ*SDVLU
      AWISTO = ADVLUF + FWSZ*SDVLUF
C
      ATADWA = AWISTO + FWSZ*SWISTO
      ATADWB = ATADWA + HWSZ*STADWA
      ATADWC = ATADWB + HWSZ*STADWB
      ATSRA  = ATADWC + HWSZ*STADWC
      ATSRB  = ATSRA  + HWSZ*STSRA
      ATIWA  = ATSRB  + HWSZ*STSRB
      ATSWKA = ATIWA
      ATIWB  = ATSWKA + HWSZ*STSWKA
      ATSWKB = ATIWB
      ATBTA  = ATSWKB + HWSZ*STSWKB
      ATBTB  = ATBTA  + HWSZ*STBTA
      ATBTC  = ATBTB  + HWSZ*STBTB
      ATWDWA = ATBTC  + HWSZ*STBTC
      ATWDWB = ATWDWA + HWSZ*STWDWA
C
      ARWDWA = ATWDWB + HWSZ*STWDWB
      ARWDWB = ARWDWA + HWSZ*SRWDWA
      ARWDWC = ARWDWB + HWSZ*SRWDWB
      ARBTA  = ARWDWC + HWSZ*SRWDWC
      ARBTB  = ARBTA  + HWSZ*SRBTA
      ARSRA  = ARBTB  + HWSZ*SRBTB
      ARIWA  = ARSRA
```



```
      ARSRB  = ARSRA  + FWSZ*SRSRA
      ARIWB  = ARSRB
      ARRHDA = ARSRB  + FWSZ*SRSRB
      ARSWKA = ARRHDA
      ARRHDB = ARSWKA + HWSZ*SRSWKA
      ARSWKB = ARRHDB
      ARSWKC = ARSWKB + HWSZ*SRSWKB
      ARDABA = ARSWKC + HWSZ*SRSWKC
      ARDABB = ARDABA + HWSZ*SRDABA
      TOP1   = ARDABB + HWSZ*SRDABB
C
C
C BUS2 BUFFERS:
C
      BASE2  = 0.
      ATHAM0 = BASE2
      ATHAM1 = ATHAM0 + FWSZ*STHAM0
      ATWE1  = ATHAM1 + FWSZ*STHAM1
      ATUO   = ATWE1  + FWSZ*STWE1
C
      ARWH   = ATUO   + FWSZ*STUO
      ARRH   = ARWH   + FWSZ*SRWH
      ARRH0  = ARRH   + FWSZ*(SRRH/2.)
      TOP2   = ARRH0  + FWSZ*SRRH0
C
C
C BUS3 BUFFERS:
C
      BASE3  = 0.
      ATONES = BASE3
      ATSP1  = ATONES + FWSZ*1.
      ATSP0  = ATSP1  + FWSZ*(STSP1/2.)
      ATSP   = ATSP1  + FWSZ*(STSP1-STSP)
      ATWSP  = ATSP1  + FWSZ*STSP1
      ATWSPZ = ATWSP
      ATCOST = ATWSPZ + FWSZ*STWSPZ
      ATXSP  = ATCOST + FWSZ*STCOST
      ATXSPR = ATXSP
      ATXSP1 = ATXSP  + FWSZ*1.
      ATPSP  = ATXSP  + DWSZ*(STXSP+1.)
      ATPSPR = ATPSP
      ATPSP1 = ATPSP  + FWSZ*1.
      ATRP   = ATPSP  + DWSZ*(STPSP+1.)
      ATC    = ATRP   + FWSZ*STRP
      ATRPP  = ATC    + FWSZ*STC
      ATWORK = ATRPP  + FWSZ*STRPP
      ATIC   = ATWORK + FWSZ*STWORK
      ATAWR  = ATIC   + HWSZ*EVEN(STIC)
      ATFILT = ATAWR  + FWSZ*1.
      ATAW   = ATFILT
      ATCW   = ATAW   + FWSZ*STAW
```

```
C  <UCA16>BBN16C.FOR.1   Mon 10-Nov-80 6:32PM          Page 1:14

        ATANS  = ATCH   + FWSZ*STCH
        ATE1   = ATFILT + FWSZ*STFILT
        ATE10  = ATE1   + FWSZ*(STE1-STE10)
        ATE11  = ATE1   + FWSZ*(STE1-STE11)
        ATRS   = ATE1   + FWSZ*STE1
        ATLMU  = ATRS   + FWSZ*STRS
        ATRSP  = ATLMU  + FWSZ*STLMU
        ATKE   = ATRSP  + FWSZ*STRSP
        ATKH   = ATKE   + FWSZ*STKE
        ATIK   = ATKH   + FWSZ*STKH
        ATE2   = ATIK   + HWSZ*EVEN(STIK)
        ATFAC  = ATE2   + FWSZ*STE2
        ATGFAC = ATFAC  + FWSZ*STFAC
        ATIDG  = ATGFAC + FWSZ*STGFAC
        ATQ1   = ATIDG  + HWSZ*EVEN(STIDG)
        ATVH   = ATQ1   + FWSZ*STQ1
        ATRH   = ATVH   + FWSZ*STVH
        ATRHB  = ATRH   + FWSZ*(STRH/2.)
C
        ARDGH  = ATRH   + FWSZ*STRH
        ARCH   = ARDGH  + FWSZ*SRDGH
        ARKH   = ARCH   + FWSZ*SRCH
        ARVHH  = ARKH   + FWSZ*SRKH
        ARVH   = ARVHH  + FWSZ*SRVHH
        TOPJ   = ARVH   + FWSZ*SRVH
C
C
C
C CONFIGURE SNAP-II BUFFERS
C
C BUS1 BUFFERS:
C
        CALL MPCLB(IBUS1+TSRA, ATSRA, STSRA, RL,   CNTG,SHRT)
        CALL MPCLB(IBUS1+TSRB, ATSRB, STSRB, RL,   CNTG,SHRT)
        CALL MPCLB(IBUS1+TSHKA,ATSHKA,STSHKA,FXD,  CNTG,LNG )
        CALL MPCLB(IBUS1+TSHKB,ATSHKB,STSHKB,FXD,  CNTG,LNG )
C
        CALL MPCLB(IBUS1+RSRA, ARSRA, SRSRA, RL,   CNTG,LNG )
        CALL MPCLB(IBUS1+RSRB, ARSRB, SRSRB, RL,   CNTG,LNG )
        CALL MPCLB(IBUS1+RSHKA,ARSHKA,SRSHKA,RL,   CNTG,SHRT)
        CALL MPCLB(IBUS1+RSHKB,ARSHKB,SRSHKB,RL,   CNTG,SHRT)
C
C
C BUS2 BUFFERS:
C
        CALL MPCLB(IBUS2+THAM0,ATHAM0,STHAM0,RL,   CNTG,LNG )
        CALL MPCLB(IBUS2+THAM1,ATHAM1,STHAM1,RL,   CNTG,LNG )
        CALL MPCLB(IBUS2+TWE1, ATWE1, STWE1, RL,   CNTG,LNG )
        CALL MPCLB(IBUS2+TWD,  ATWD,  STWD,  RL,   CNTG,LNG )
C
        CALL MPCLB(IBUS2+RWH,  ARWH,  SRWH,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS2+RRH,  ARRH,  SRRH,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS2+RRHB, ARRHB, SRRHB, RL,   CNTG,LNG )
```



```
C  <UCA16>BBN16C.FOR.1   Mon 10-Nov-80 6:32PM          Page 1:15

C
C
C BUS3 BUFFERS:
C
        CALL MPCLB(IBUS3+TUNES,ATUNES,STUNES,RL,   0   ,LNG )
        CALL MPCLB(IBUS3+TSP1, ATSP1, STSP1, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TSP0, ATSP0, STSP0, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TSP,  ATSP,  STSP,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TWSP, ATWSP, STWSP, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TWSPZ,ATWSPZ,STWSPZ,RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TCOST,ATCOST,STCOST,RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TXSP, ATXSP, STXSP, CMPLX,CNTG,LNG )
        CALL MPRPB(TXSPR,TXSP)
        CALL MPSBS(TXSPR,1)
        CALL MPIPB(TXSPI,TXSP)
        CALL MPSBS(TXSPI,1)
        CALL MPCLB(IBUS3+TPSP, ATPSP, STPSP, CMPLX,CNTG,LNG )
        CALL MPRPB(TPSPR,TPSP)
        CALL MPSBS(TPSPR,1)
        CALL MPIPB(TPSPI,TPSP)
        CALL MPSBS(TPSPI,1)
        CALL MPCLB(IBUS3+TRP,  ATRP,  STRP,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TC,   ATC,   STC,   RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TRPP, ATRPP, STRPP, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TWORK,ATWORK,STWORK,RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TIC,  ATIC,  STIC,  FXD,  CNTG,LNG )
        CALL MPCLB(IBUS3+TFILT,ATFILT,STFILT,RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TAH,  ATAH,  STAH,  RL,   CNTG,LNG )
        CALL MPCRB(TAHR,TAH)
        CALL MPSBS(TAHR,1)
        CALL MPCLB(IBUS3+TCH,  ATCH,  STCH,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TANS, ATANS, STANS, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TE1,  ATE1,  STE1,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TE10, ATE10, STE10, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TE11, ATE11, STE11, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TRS,  ATRS,  STRS,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TLMU, ATLMU, STLMU, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TRSP, ATRSP, STRSP, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TKE,  ATKE,  STKE,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TKH,  ATKH,  STKH,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TIK,  ATIK,  STIK,  FXD,  CNTG,LNG )
        CALL MPCLB(IBUS3+TE2,  ATE2,  STE2,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TFAC, ATFAC, STFAC, RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TGFAC,ATGFAC,STGFAC,RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TIDG, ATIDG, STIDG, FXD,  CNTG,LNG )
        CALL MPCLB(IBUS3+TQ1,  ATQ1,  STQ1,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TVH,  ATVH,  STVH,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TRH,  ATRH,  STRH,  RL,   CNTG,LNG )
        CALL MPCLB(IBUS3+TRHB, ATRHB, STRHB, RL,   CNTG,LNG )
```

```
C
       CALL MPCLB(IBUS3+RDGN, ARDGN, SRDGN, RL,   CNTG,LNG )
       CALL MPCLB(IBUS3+RCN,  ARCN,  SRCN,  RL,   CNTG,LNG )
       CALL MPCLB(IBUS3+RKN,  ARKN,  SRKN,  RL,   CNTG,LNG )
       CALL MPCLB(IBUS3+RVNN, ARVNN, SRVNN, RL,   CNTG,LNG )
       CALL MPCLB(IBUS3+RVN,  ARVN,  SRVN,  RL,   CNTG,LNG )
C
C
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CONFIGURE SNAP-II REAL SCALARS:
C
C THIS MODULE CONFIGURES THE SYSTEM REAL SCALARS.
C ALL MAP REAL SCALAR ID'S (SID'S) ARE SYMBOLICALLY
C NAMED, WITH NAMES STARTING WITH 'T' FOR
C TRANSMITTER SCALARS, AND NAMES STARTING WITH 'R'
C FOR RECEIVER SCALARS.
C
C
C
C PREEMPHASIS FILTER COEFFICIENT
       TALPH = 50
C
C PREEMPHASIS FILTER HISTORY
       TPHST = 51
C
C NEGATIVE OF CURRENT FRAME D.C. TERM
       TDCN  = 52
C
C NEGATIVE OF RECIPROCAL OF SIZE OF PITCH EXTRACTION BUFFER
       TFSZI = 53
C
C (QUANTIZED) PITCH LAG (IN SAMPLES)
       TNH   = 54
C
C CODED PITCH LAG (INTEGER IN LEFT HALFWORD)
       TIN   = 55
C
C CONSTANT 14
       T14   = 56
C
C THRESHOLD VALUE FOR NEWLP, NULOG
       TKTHR = 57
C
C CODED FRAME ENERGY (INTEGER IN LEFT HALFWORD)
       TIG   = 58
C
C PITCH PREDICTION COEFF FOR SINGLE TAP CASE
       TC2P  = 59
```



```
C
C CONSTANT RECIPROCAL OF 72
       T72R  = 60
C
C CONSTANT RECIPROCAL OF 3
       T3R   = 61
C
C RECEIVED SQRT(FRAME ENERGY)
       RGH   = 65
C
C RECEIVED PITCH
       RNH   = 66
C
C DEEMPHASIS FILTER COEFFICIENT SA0 (MUST BE CONTIGUOUS /---\ )
       RDCF0 = 67
C                                                         !
C DEEMPHASIS FILTER COEFFICIENT SA1                       !
       RDCF1 = 68
C                                                         !
C DEEMPHASIS FILTER COEFFICIENT SA2                       !
       RDCF2 = 69
C                                                         !
C DEEMPHASIS FILTER COEFFICIENT SA3                       !
       RDCF3 = 70
C                                                     ( \---/ )
C DEEMPHASIS FILTER MEMORY Y(-2)    (MUST BE CONTIGUOUS /---\ )
       RDMY0 = 71
C                                                         !
C DEEMPHASIS FILTER MEMORY Y(-1)                          !
       RDMY1 = 72
C                                                         !
C DEEMPHASIS FILTER MEMORY U(-2)                          !
       RDMY2 = 73
C                                                         !
C DEEMPHASIS FILTER MEMORY U(-1)                          !
       RDMY3 = 74
C                                                     ( \---/ )
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CONFIGURE SNAP-II INTEGER SCALARS
C
C THIS MODULE CONFIGURES THE MAP INTEGER SCALARS.
C ALL MAP INTEGER SCALAR ID'S (ISID'S) ARE
C SYMBOLICALLY NAMED, WITH NAMES (IN GENERAL) STARTING WITH
C 'T' FOR TRANSMITTER SCALARS, AND
C WITH 'R' FOR RECEIVER SCALARS.
C (FOR BUFFER STATUS FLAGS: 0=>EMPTY, 1=>FULL)
C
```

```
C
C
C
C TSRA BUFFER STATUS FLAG
        TSRFA = 50
C
C TSRB BUFFER STATUS FLAG
        TSRFB = 51
C
C TBTA BUFFER STATUS FLAG
        TBTFA = 52
C
C TBTB BUFFER STATUS FLAG
        TBTFB = 53
C
C NEWLP OUTPUT: NUMBER OF ITERATIONS NOT COMPLETED (TINC,TFENG CONTIG.)
        TINC  = 54
C
C NEWLP OUTPUT: FRACTIONAL ENERGY E(N)/R(0)       (TINC,TFENG CONTIG.)
        TFENG = 55
C
C (UNUSED)
        I     = 56
C
C RBTA BUFFER STATUS FLAG
        RBTFA = 57
C
C RBTB BUFFER STATUS FLAG
        RBTFB = 58
C
C RSMKA BUFFER STATUS FLAG
        RSMFA = 59
C
C RSMKB BUFFER STATUS FLAG
        RSMFB = 60
C
C SYSTEM RUN FLAG (0 => STOP)
        RUN   = 61
C
C (LEFT UNUSED FOR NPITH)
        I     = 62
C
C (UNUSED)
        I     = 63
C
C TADBA/B BUFFER POINTER (0=>B; -2=>A)
        ADPO  = 64
C
C TSRA/B BUFFER POINTER (0=>B; -2=>A)
        TSRPO = 65
C
```



```
C TBTA/B BUFFER POINTER (0=>B; -2=>A)
        TBTPO = 66
C
C TMDMA/B BUFFER POINTER (0=>B; -2=>A)
        TMPO  = 67
C
C RMDMA/B/C BUFFER POINTER (0=>C; -2=>B; -4=>A)
        RMPO  = 68
C
C RBTA/B BUFFER POINTER (0=>B; -2=>A)
        RBTPO = 69
C
C RSMKA/B BUFFER POINTER (0=>B; -2=>A)
        RSMPO = 70
C
C RDABA/B BUFFER POINTER (0=>B; -2=>A)
        DAPO  = 71
C
C A/D FRAME DISCARD COUNTER
        TADDC = 72
C
C TMODEM FAKE FRAME COUNTER
        TMFFC = 73
C
C RMODEM FRAME DISCARD COUNTER
        RMFDC = 74
C
C RSINK DATA-NOT-READY COUNTER
        RSNNR = 75
C
C TRANSMITTER FRAME COUNTER
        TFCTR = 76
C
C RECEIVER FRAME COUNTER
        RFCTR = 77
C
C RECEIVER LOST-SYNC COUNTER
        RLCTR = 78
C
C LOCAL HANDSET ON-HOOK STATE (0=>HANDSET OFF-HOOK)
        RONHK = 79
C
C STATE OF RECEIVER SYNC (0=>LOST SYNC; -1=>SRCHING SYNC; 1=>HAVE SYNC.
        RSYNC = 80
C
C BEGINNING-OF-FRAME OFFSET (SYNC BIT POSITION IN RMODEM BUFFER)
        RBOFO = 81
C
C (LEFT UNUSED FOR NPITH)
        I     = 82
C
```

```
C  <UCA16>MMM16C.FOR.1    Mon 18-Nov-88 6:32PM          Page 1:28

C A/D (ADAM) INTERRUPT COUNTER
        ADCTR = 03
C
C TMODEM (IOS-2) INTERRUPT COUNTER
        TMCTR = 04
C
C RMODEM (IOS-2) INTERRUPT COUNTER
        RMCTR = 05
C
C D/A (AOM) INTERRUPT COUNTER
        DACTR = 06
C
C A/D PEAK INPUT LEVEL
        TADPK = 07
C
C NO-ERROR-CORRECTION SWITCH (NO CORRECTIONS IF NZ)
        RNCOR = 123
C
C (LEFT UNUSED FOR MPITH)
        I     = 124
C
C CHANNEL ERROR SIMULATOR FLAG (NON-ZERO => # SIM ERRORS PER FRAME)
        RERRS = 125
C
C (LEFT UNUSED FOR MPITH)
        I     = 126
C
C
C
C
C
        RETURN
        END
```

```
C (BBNOI)<DCA16>BBN16D.FOR.1, 10-Nov-80 10:58:33, Ed: KFIELD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       BBN16D - DUMMY ROUTINES FOR UNSUPPORTED
C               FILE-TO-FILE AND TIMING MODES OF
C               OPERATION.
C
C       INCLUDES DUMMY ROUTINES FOR:
C               IDELT(I,J)
C               IRDCLK(I)
C               SETCLK(I)
C               SCOPY(IARRAY,JARRAY)
C               CNFIRM(IARRAY)
C               KOPE(IARRAY,J,K)
C               KSIN(IARRAY,JARRAY,K,L)
C               KOPN(IARRAY,J,K,LOG)
C               KSOUT(IARRAY,JARRAY,K,L)
C               KCLOSE(IARRAY)
C               DELAY(I,J)
C
C KFIELD 11/80
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
        INTEGER FUNCTION IDELT(I,J)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION IRDCLK(I)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        SUBROUTINE SETCLK(I)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        SUBROUTINE SCOPY(IARRAY,JARRAY)
        DIMENSION IARRAY(1),JARRAY(1)
        TYPE 100
```



```
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        LOGICAL FUNCTION CNFIRM(IARRAY)
        DIMENSION IARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KOPE(IARRAY,J,K)
        DIMENSION IARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KSIN(IARRAY,JARRAY,K,L)
        DIMENSION IARRAY(1),JARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KOPN(IARRAY,J,K,LOG)
        DIMENSION IARRAY(1)
        LOGICAL LOG
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        INTEGER FUNCTION KSOUT(IARRAY,JARRAY,K,L)
        DIMENSION IARRAY(1),JARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
```

```
C  <OCA16>NBN16D.FOR.1   Mon 18-Nov-85 6:58PM          Page 1:2

C
        INTEGER FUNCTION KCLOSE(IARRAY)
        DIMENSION IARRAY(1)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
C
C
C
        SUBROUTINE DELAY(I,J)
        TYPE 100
 100    FORMAT(' THIS MODE OF OPERATION NOT YET FULLY IMPLEMENTED.')
        STOP
        END
```

```
C [BBND]<DCA16>BBN16E.FOR.1, 18-Nov-88 18:35:10, Ed: KFIELD
      SUBROUTINE BBN16E
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                       BBN16E
C
C BBN16 SPEECH CODER SYSTEM EXECUTION MODULE
C
C
C       EXECUTION MODE ENTERED BY USER:
C       1) REAL TIME SYSTEM
C       2) FILE TO FILE SYSTEM
C       3) TIMING SYSTEM
C       4) SCOPE TIMING SYSTEM
C
C CALLED BY BBN16R
C
C KFIELD 9/80
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C       HOST ARRAYS FOR FILE TO FILE TEST SYSTEM
C
        INTEGER XSIN(264),XSOT(264)
        INTEGER MORBLK(256),MSG(20)
        INTEGER ISIG(230),IST(87)
        REAL SIG(230)
        EQUIVALENCE (MDRBLK(1),MSG(1),SIG(1),IST(1))
        LOGICAL CNFIRM,LTSMK,LRIST,ABUFS,BBUFS
C
C       MAP FUNCTION LIST NAMES
C
        INTEGER BBNRTS,BBNLP,BBNFFT,BBNTIM,BBNRST
C
C       OTHER MAP-RELATED VARIABLES
C
        REAL HWSZ,FWSZ,DWSZ
        INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
        INTEGER PRCSR,AP,ME,MIS
C
        LOGICAL*1 CHRC,CHRE,CHRL,CHRR,CHRS,CHRT,CHRQ,CHRUM,ICMD
        REAL MDMPRG,ADMPRG,AOMPRG
        INTEGER MDMSCL,ADAM,AOM,IOS2
C
C
C
C
C [MMMD]<MAP>BBN16-COMMON.FOR.12, 31-Oct-80 23:17:03, Ed: WOLF
C
C
```



```
C MAP BUFFER NAMES:
C
        INTEGER DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3
        INTEGER DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU
        INTEGER DVLUF
        INTEGER CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3
        INTEGER CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU
        INTEGER HISTO
        INTEGER TADBA,TADBB,TADBC,TSWA, TSWB, TOWEJ
        INTEGER TSP0, TSP1, TSP,  THAMB,TWSP, TWSPZ
        INTEGER TCOST,TXSP, TXSPW,TXSPI,TPSP, TPSPW
        INTEGER TPSPI,TRP,  TC,   TWORK,TIC,  TCH
        INTEGER TE10, TE11, TE1,  THAM1,TWE1, TRS
        INTEGER TLMU, TRSP, TKE,  TKH,  TIK,  TFILT
        INTEGER TAN,  TANR, TE2,  TFAC, TANS, TGFAC
        INTEGER TIDG, TUU,  TUI,  TVU,  TRW0, TRW
        INTEGER TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB
        INTEGER TBTC, TMDMA,TMDMB,TRPP
        INTEGER RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSRA
        INTEGER RSWB, RUMA, RUMB, RDGH, RCH,  RKH
        INTEGER RWH,  RVHM, RVH,  RRH0, RRH,  RRHDA
        INTEGER RRHDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C MAP REAL SCALAR NAMES:
C
        INTEGER TALPH,TPHST,TDCN, TFSZI,TNH,  TIM
        INTEGER T14,  TKTHR,TIG,  TC2P, T72R, T3K
        INTEGER RGN,  RMH,  RDCF0,RDCF1,RDCF2,RDCF3
        INTEGER RDMY0,RDMY1,RDMY2,RDMY3
C
C
C MAP INTEGER SCALAR NAMES:
C
        INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG
        INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RUN
        INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO, RBTPO
        INTEGER RSNPO,DAPO, TADDC,TMFFC,RMFDC,RSNMR
        INTEGER TFCTR,RFCTR,RLCTR,RONHK,RSYNC,RBUFO
        INTEGER ADCTR,TMCTR,RMCTR,DACTR,TAUPK
        INTEGER RNCOR,RERRS
C
C
C COMMON BLOCK - BUFFER AND SCALAR NAMES:
C
        COMMON /BSIDS/
     )  DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3,
     )  DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU,
     )  DVLUF,
     )  CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3,
     )  CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU,
```

```
     )  HISTO,
     )  TADBA,TADBB,TADBC,TSMA, TSMB, TOMES,
     )  TSP0, TSP1, TSP,  THAMB,TMSP, TMSPZ,
     )  TCOST,TKSP, TKSPR,TKSPI,TPSP, TPSPR,
     )  TPSPI,TRP,  TC,   TBMK, TIC,  TCM,
     )  TE10, TE11, TE1,  THAM1,TME1, TRS,
     )  TLMB, TRSP, TKE,  TKM,  TIK,  TFILT,
     )  TAM,  TAMR, TK2,  TFAC, TAMS, TGFAC,
     )  TIUG, TWU,  TQ1,  TVM,  TRHB, TRH,
     )  TIUA, TIUB, TSMKA,TSMKB,TBTA, TBTB,
     )  TBTC, TMDMA,TMDMB,TRPP,
     )  RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSMA,
     )  RSRB, RUHA, RUHB, ROGH, RCH,  RKH,
     )  RMM,  RVMM, RVM,  RRMB, RRM,  RRMDA,
     )  RRMDB,RSMKA,RSMKB,RSMKC,RDAMA,RDAMB,
     )  TALPH,TPHST,TDCM, TFSZI,TMH,  TIM,
     )  T14,  TKTHR,TIG,  TC2P, T72R, T3R,
     )  RGM,  RNH,  RDCF0,RDCF1,RDCF2,RDCF3,
     )  RDMY0,RDMY1,RDMY2,RDMY3,
     )  TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG,
     )  RBTFA,RBTFB,RSMFA,RSMFB,RUM,
     )  ADPU, TSMPO,TMTPU,TMPO, RMPO, RMTPU,
     )  RSMPO,DAPO, TADDC,TMFFC,RMFDC,RSMMR,
     )  TFCTR,RFCTR,RLCTR,RDMMK,RSYNC,RBUFO,
     )  ADCTR,TMCTR,RMCTR,DACTR,TADPK,
     )  RMCOM,MEMRS
C
C
C COMMON BLOCK - BUFFER ADDRESSES AND SIZES
C        NEEDED FOR INIT OF NON-SNAP BUFFERS
C        AND FOR FILE-TO-FILE MODE OF OPERATION
C
        COMMON /BAS/
     )  AHISTO,SHISTO,
     )  ATADBA,STADBA,
     )  ATADBB,STADBB,
     )  ATADBC,STADBC,
     )  ATBTC, STBTC,
     )  ATMDMA,STMDMA,
     )  ATMDMB,STMDMB,
     )  ARMDMA,SRMDMA,
     )  ARMDMB,SRMDMB,
     )  ARMDMC,SRMDMC,
     )  ARSMKC,SRSMKC,
     )  ARDAMA,SRDAMA,
     )  ARDAMB,SRDAMB
C
CCCCCCC END OF BBN16-COMMON CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       COMMON BLOCK - FUNCTION LIST NAMES
```



```
C
        COMMON /FLS/
     )BBNRTS,BBNLP,BBNFFT,BBNTIM,BBNHST
C
C
C
        DATA PRCSR/1/,AP/1/,ME/1/,HIS/2/
        DATA NL/0/,CMPLX/1/,FXU/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA HWSZ/1./,FWSZ/2./,DWSZ/4./
        DATA IBUS1/64/,IBUS2/128/,IBUS3/192/
C
        DATA CHRC,CHRE,CHRL,CHRR,CHRS,CHRT/'C','E','L','R','S','T'/
        DATA CHRQ,CHRQM/'Q','?'/
C  SCROLL IDENTIFIERS
        DATA MONSCL,ADAM,AQM,IOS2/16,23,22,2/
C  BUS 1 ADDRESSES OF SCROLL PROGRAMS
        DATA MONPRG,ADMPRG,AQMPRG/18432.,18592.,18656./
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C!!!!!!! FORCE MODE TO INDICATE 'REAL TIME SYSTEM' !!!!!!!!
C (OTHER MODES NOT FULLY SUPPORTED IN FINAL SYSTEM.)
C
        MODE = 1
        IF (MODE .EQ. 1) GOTO 100
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       GET EXECUTION MODE FROM USER...
C
   9    TYPE 10
  10    FORMAT(' EXECUTION MODES:'/
     )'    1 => REAL TIME SYSTEM'/
     )'    2 => FILE TO FILE SYSTEM'/
     )'    3 => TIMING SYSTEM'/
     )'    4 => SCOPE TIMING SYSTEM'/
     )' ENTER EXECUTION MODE: '$)
        ACCEPT 11,MODE
  11    FORMAT(I3)
C
        IF ((MODE .LT. 1) .OR. (MODE .GT. 4)) GOTO 9
C
C       EXECUTE CODE FOR INDICATED MODE (DO MODE CODE)
C
        IF (MODE .EQ. 1) GOTO 100
        IF (MODE .EQ. 2) GOTO 200
        IF (MODE .EQ. 3) GOTO 300
        IF (MODE .EQ. 4) GOTO 400
```

```
C  <DCA16>BBN16E.FOR.1   Mon 10-Nov-80 6:35PM        Page 1:4

C
C
C       ALL MODES RETURN HERE
C
 900    CONTINUE
C
        RETURN
```

```
C  <DCA16>BBN16E.FOR.1   Mon 10-Nov-80 6:35PM        Page 2

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE BBN16 REAL TIME SYSTEM
C
 100    CONTINUE
C
C  CONFIGURE A BUFFER OVER EACH SCROLL PROGRAM AND LOAD IT
C
        CALL MPCLB(IBUS1+63,MDMPRG,512.,FXD,CNTG,LNG)
        CALL MPLDS(MDMSCL,IOS2,63)
        CALL MPCLB(IBUS1+63,ADMPRG,512.,FXD,CNTG,LNG)
        CALL MPLDS(ADAM,IOS2,63)
        CALL MPCLB(IBUS1+63,ADMPRG,512.,FXD,CNTG,LNG)
        CALL MPLDS(ADM,IOS2,63)
C
C
C
C  START THE REAL TIME SYSTEM
C
        CALL MPXFL(BBNNTS)
C
        TYPE 100
100     FORMAT(' SPEECH CODER IS IN OPERATION.'/)
110     TYPE 111
111     FORMAT(' COMMANDS ARE:'/
        1 '   S:   SUSPEND TASK (LEAVING SPEECH CODER RUNNING)'/
        1 '   Q:   QUIT (HALT SPEECH CODER)'/
        2 '   E N: SIMULATE N ERRORS PER FRAME'/
        3 '   C N: ERROR-CORRECT IF N=0'/
        4 '   L:   CAUSE RCVR TO LOSE SYNC'/
        5 '   T:   TYPE OUT SPEECH CODER STATE')
112     TYPE 113
113     FORMAT(/' ENTER COMMAND: '$)
        ACCEPT 114,ICMD,N
114     FORMAT(A1,I)
        IF(ICMD.EQ.CHRQN) GO TO 110
        IF(ICMD.EQ.CHRQ) GO TO 120
        IF(ICMD.EQ.CHRE) GO TO 130
        IF(ICMD.EQ.CHRC) GO TO 140
        IF(ICMD.EQ.CHRL) GO TO 150
        IF(ICMD.EQ.CHRT) GO TO 160
        IF(ICMD.EQ.CHRS) GO TO 170
115     TYPE 116
116     FORMAT(' BAD COMMAND!')
        GO TO 110
```

```
C  "Q" COMMAND -- QUIT (HALT SPEECH CODER)
120     CALL MPITM(RUN,0,0)
        CALL MPSAA(ADAM)
        CALL MPSAA(ADM)
C  (THERE'S NO WAY TO STOP THE MODEM SCROLL!)
C
C       RETURN TO COMMON CODE
        GO TO 900
C
C "E" COMMAND -- SET ERROR SIMULATION
130     IF(N.LT.0 .OR. N.GT.50) GO TU 115
        CALL MPITM(RERRS,N,0)
        GO TO 112
C
C  "C" COMMAND -- CONTROL ERROR CORRECTION
140     IF(N.NE.0) N=1
        CALL MPITM(RNCOR,N,0)
        GO TO 112
C
C  "L" COMMAND -- CAUSE THE RCVR TO LOSE SYNC BY JAMMING INTO RBOFO
150     CALL MPITM(RBOFO,50,0)
        GO TU 112
C
C "T" COMMAND -- STOP CODER, READ INTEGER SCALARS, RESET TADPK
C    (PEAK INPUT LEVEL SO FAR) TO ZERO, THEN RESTART,
C    AND TYPE OUT INTEGER SCALAR VALUES
160     CALL MPITM(RUN,0,0)
        CALL MPIRS(TSRFA,IST(TSRFA),30,0)
        CALL MPIST(TADPK,0)
        CALL MPXFL(BBNNST)
        CALL TIST(IST)
        GO TO 112

C  "S" COMMAND -- SUSPEND THIS TASK, BUT LET MAP CONTINUE.
C  (UPON RESUMPTION OF TASK, IT WILL CONTINUE OUT OF THE PAUSE)
170     PAUSE '(SPEECH CODER CONTINUING)'
        GO TU 110
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE FILE TO FILE SYSTEM
C
 200    CONTINUE
C
C
C
C       DEFINE INPUT AND OUTPUT FILES
C
 203    TYPE 210
 210    FORMAT(' READ INPUT SIGNAL FROM FILE: '$)
        NBLKS = KOPE(KSIN,0,0)
        IF(NBLKS.LE.0) GO TO 203
C
        CALL SCOPY('SKIP HEADER BLOCK (256 WORDS)? ',MSG)
        IF (.NOT. CNFIRM(MSG)) GOTO 215
        ISTAT = KSIN(KSIN,WORBLK,256,NWXFRD)
C
 215    TYPE 220
 220    FORMAT(' WRITE OUTPUT SIGNAL TO FILE: '$)
        NBLKS = KOPN(KSOT,0,0,.TRUE.)
        IF(NBLKS.LE.0) GO TO 215
C
C       USE DUMMY BUFFER TO READ & WRITE TADBA,B & RDABA,B ,
C       SINCE THEY ARE RL,SHORT FORMAT, AND MPWDB/MPRDB
C       DON'T PROPERLY ACCESS RL,SHORT BUFFERS.
C       USE BUFFER FROM SYSTEM WHICH IS CORRECT LENGTH (216
C       SAMPLES), RL, LNG, AND IS NOT NEEDED FOR FRAME-TO-
C       FRAME MEMORY.
C
        IDUM = TAE1
        IADBUF = 60
        IDABUF = 61
        ITMDM  = 60
        IRMDM  = 61
C
C INITIALIZE SYNC BIT OF RMDMA TO 1 FOR FIRST RMINT CALL
C (BY USING INITIALIZED TMDMB BUFFER), SO 2ND RMINT CALL
C (WITH VALID DATA IN RMDMB) SYNC BIT = 0) WILL SHOW
C ALTERNATING SYNC BIT.
C
        CALL MPCLM(IBUS1+ITMDM,ATMDMB,STMDMB,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRMDM,ARMDMA,SRMDMA,FXD,CNTG,LNG)
        CALL VMOV(IRMDM,ITMDM)
C
        IEOF = -1
        NFRAME=0
        NCLIPS=0
        CALL MPIST(RUN,1)
```

```
C  <DCA16>BBN16E.FOR.1   Mon 10-Nov-80 6:35PM          Page 3:1

C  ASK ABOUT FRAME TYPEOUTS
        LTSNK=.FALSE.
        LRIST=.FALSE.
        TYPE 226
226     FORMAT(' FRAME TYPEOUTS? ("T"SNK, "S"CALARS, OR NIL): '$)
        ACCEPT 227,ICMD
227     FORMAT(A1)
        IF(ICMD.EQ.CHRS) LRIST=.TRUE.
        IF(ICMD.NE.CHRT) GO TO 230
        LTSNK=.TRUE.
        TYPE 228
228     FORMAT(' FILE FOR TSINK DATA: '$)
        CALL ASSIGN(2,XSIN,-1)
        WRITE(2,229)
229     FORMAT(/'  FN#    G UG1 UG2 UG3    N  C1  C2  C3  K1  ',
     1  'K2  K3  K4  K5  K6  RESIDUAL'/)
C
C  TOP OF LOOP
C
230     ISTAT = KSIN(KSIN,ISIG,216,NWKFRD)
        IF (ISTAT .EQ. IEOF) GOTO 290
        DO 235 I=1,216
235     SIG(I) = FLOAT(ISIG(I)) / 32768.
C
C  SET FLAG FOR "A" OR "B" BUFFERS
        ABUFS = .FALSE.
        IF(MOD(NFRAME,2).EQ.0) ABUFS = .TRUE.
        BBUFS = .NOT. ABUFS
C
        IF (ABUFS) ITSNK=TSNKA
        IF (BBUFS) ITSNK=TSNKB
C
C       XFER SIG TO MAP
C
        CALL MPWDB(IDUM,SIG,4,1,SIG(216))
        IF (ABUFS) CALL MPCLB(IBUS1+IADBUF,ATADBA,STADBA,RL,CNTG,SHRT)
        IF (BBUFS) CALL MPCLB(IBUS1+IADBUF,ATADBB,STADBB,RL,CNTG,SHRT)
        CALL MPWT(PRCSR,NIS)
        CALL VMOVB(IADBUF,IDUM)
        CALL MPWT(PRCSR,AP)
C
C
C       EXECUTE 1 PASS THRU SYSTEM LOOP
C
C
        CALL MPXFL(BBNFFT)
C
C MAKE MUST WAIT 12*16.67 MSEC = 200 MSEC.
        CALL DELAY(2,12)
C
C
C  NOW MOVE BITS FROM THODEN BUFFER TO RNODEN BUFFER, AS A CHANNEL
C    DOES.
        GO TO(241,242,243,244,245,246) 1+MOD(NFRAME,6)
```

```
C  <DCA16>BBN16E.FOR.1   Mon 10-Nov-80 6:35PM          Page 3:2

241     CALL MPCLB(IBUS1+ITNDN,ATNDNA,STNDNA,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRNDN,ARNDNB,SRNDNB,FXD,CNTG,LNG)
        GO TO 248

242     CALL MPCLB(IBUS1+ITNDN,ATNDNB,STNDNB,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRNDN,ARNDNC,SRNDNC,FXD,CNTG,LNG)
        GO TO 248

243     CALL MPCLB(IBUS1+ITNDN,ATNDNA,STNDNA,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRNDN,ARNDNA,SRNDNA,FXD,CNTG,LNG)
        GO TO 248

244     CALL MPCLB(IBUS1+ITNDN,ATNDNB,STNDNB,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRNDN,ARNDNB,SRNDNB,FXD,CNTG,LNG)
        GO TO 248

245     CALL MPCLB(IBUS1+ITNDN,ATNDNA,STNDNA,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRNDN,ARNDNC,SRNDNC,FXD,CNTG,LNG)
        GO TO 248

246     CALL MPCLB(IBUS1+ITNDN,ATNDNB,STNDNB,FXD,CNTG,LNG)
        CALL MPCLB(IBUS1+IRNDN,ARNDNA,SRNDNA,FXD,CNTG,LNG)
        GO TO 248
C
248     CALL VMOV(IRNDN,ITNDN)
C
C       READ OUTPUT AND WRITE TO FILE
C
C
        IF (ABUFS) CALL MPCLB(IBUS1+IDABUF,ARDABA,SRDABA,RL,CNTG,SHRT)
        IF (BBUFS) CALL MPCLB(IBUS1+IDABUF,ARDABB,SRDABB,RL,CNTG,SHRT)
        CALL VMOVN(IDUN,IDABUF)
        CALL MPWT(PRCSR,AP)
C
        CALL MPRDB(IDUN,SIG,4,1,SIG(216))
C
        DO 250 I=1,216
          XSIG = SIG(I) * 32768.
          IF (ABS(XSIG) .LE. 32767.) GO TO 250
          NCLIPS=NCLIPS+1
          XSIG=SIGN(32767.,XSIG)
250       ISIG(I) = IFIX(XSIG)
C
C       WRITE OUT INTEGER OUTPUT
C
        ISTAT = KSOUT(XSOT,ISIG,216,NWKFRD)
C
C       LOOP.
C
```

```
C  IF REQUESTED, DO FRAME TYPEOUT
        IF(.NOT.LTSNK) GO TO 260
        CALL MPRDB(ITSNK,ISIG,2,0,ISIG(230))
        WRITE(2,255) NFRAME,(ISIG(I), I=217,230),(ISIG(I),I=1,216)
 255    FORMAT(I5,1X,14I4,/,72I1,/,72I1,/,72I1,/)
 260    IF(LRIST) CALL RIST
        NFRAME=NFRAME+1
        GOTO 230
C
C       HERE ON EOF.
C
 290    CALL RIST
        TYPE 295,NFRAME,NCLIPS
 295    FORMAT(I7,' FRAMES,',I5,' OUTPUT SAMPLES WERE CLIPPED.')
        ISTAT = KCLOSE(ISIN)
        ISTAT = KCLOSE(ISOT)
        IF(LTSNK) CALL CLOSE(2)
C
C
C       RETURN TO COMMON CODE
C
        GOTO 900
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE TIMING SYSTEM
C
 300    CONTINUE
C
C       GET TIMING PARAMETERS.
C
        TYPE 310
 310    FORMAT(' ENTER ITERATION VALUE: '$)
        ACCEPT 311,N
 311    FORMAT(I3)
C
C
C  PUT SOMETHING REASONABLE IN TAUBA, TAUBB
C  (DON'T USE BID 63 SINCE VCOST USES IT IMPLICITLY.)
C
        IADBUF = 60
        CALL MPCLB(IBUS1+IADBUF,ATAUBA,STAUBA,NL,CNTG,SHRT)
        CALL VCOST(IADBUF,17.)
        CALL MPCLB(IBUS1+IADBUF,ATAUBB,STAUBB,NL,CNTG,SHRT)
        CALL VCOST(IADBUF,17.)
C
C
C
C
C       SET UP TIMING (100 USEC TICKS)
C       READ CLOCK
C
        CALL SETCLK(1)
        IT1 = IRDCLK(0)
C
C       EXECUTE BBNTIN "N" TIMES
C
        CALL MPIST(NUM,1)
        CALL MPFOR(1,1,N,BBNTIN)
C
C       WAIT ON MAP, READ CLOCK
C
        CALL MPRST(1,NONE,1,1)
        IT2 = IRDCLK(0)
        TIME = FLOAT(IDELT(IT1,IT2))
C
C       CONVERT TIME, PRINT IT
C
        TIMEMS = TIME * .1
        TYPE 320,N,TIMEMS
 320    FORMAT(I6,' ITERATION(S) TOOK',F10.4,' MSECS.')
C
C       RETURN TO COMMON CODE
C
        GOTO 900
^L
```

```
C  <DCA16>BBN16E.FOR.1   Mon 10-Nov-80 6:35PM         Page 5

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       EXECUTE SCOPE TIMING SYSTEM
C
 400    CONTINUE
C
C
C  PUT SOMETHING REASONABLE IN TADBA,TADBB
C  (DON'T USE BIU 63 SINCE VCOST USES IT IMPLICITLY.)
C
        IADBUF = 60
        CALL MPCLB(IBUS1+IADBUF,ATADBA,STADBA,RL,CNTG,SHRT)
        CALL VCOST(IADBUF,17.)
        CALL MPCLB(IBUS1+IADBUF,ATADBB,STADBB,RL,CNTG,SHRT)
        CALL VCOST(IADBUF,17.)
C
C
C       START UP INFINITE MPIWL ON BBNTIM
C
        CALL MPIST(RUN,1)
        CALL MPIWL(RUN,NS,0,BBNTIM)
C
C       WAIT FOR USER TO SAY QUIT
C
        TYPE 410
 410    FORMAT(' TYPE ANY CHARACTER TO HALT MAP: '$)
        ACCEPT 411,ICHAR
 411    FORMAT(A1)
C
C       STOP MPIWL
C
        CALL MPITM(RUN,0,0)
C
C       RETURN TO COMMON CODE
C
        GOTO 900
C
C
        END
^L
```

```
C  <DCA16>BBN16E.FOR.1   Mon 10-Nov-80 6:35PM         Page 6

        SUBROUTINE RIST
C  READ IST AND TYPE OUT CONTENTS
        DIMENSION IST(87)
C
        CALL MPIWS(1,IST(1),87,0)
        CALL TIST(IST)
C
        RETURN
        END
C
        SUBROUTINE TIST(IST)
C       TYPE OUT SELECTED VALUES FROM IST ARRAY
C
C
C [BBND]<MAP>BBN16-COMMON.FOR.12, 31-Oct-80 23:17:03, Ed: WOLF
C
C
C MAP BUFFER NAMES:
C
        INTEGER DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3
        INTEGER DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU
        INTEGER DVLUF
        INTEGER CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3
        INTEGER CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU
        INTEGER HISTO
        INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, TONES
        INTEGER TSP0, TSP1, TSP,  THAM0,TWSP, TWSP2
        INTEGER TCOST,TXSP, TXSPW,TXSPI,TPSP, TPSPW
        INTEGER TPSPI,TRP,  TC,   TWORK,TIC,  TCH
        INTEGER TE10, TE11, TE1,  THAM1,TWE1, TWS
        INTEGER TLMU, TRSP, TKE,  TKW,  TIK,  TFILT
        INTEGER TAH,  TAHR, TE2,  TFAC, TANS, TGFAC
        INTEGER TIDG, TUD,  TQ1,  TVH,  TRHW, TRH
        INTEGER TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB
        INTEGER TBTC, THDNA,THDNB,TRPP
        INTEGER RNDNA,RNDNB,RNDNC,RBTA, RBTB, RSRA
        INTEGER RSRB, RUHA, RUHB, RDGH, RCH,  RKH
        INTEGER RWH,  RVWH, RVH,  RRH0, RRH,  RRHDA
        INTEGER RRHDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C MAP REAL SCALAR NAMES:
C
        INTEGER TALPH,TPHST,TDCH, TFSZ1,TMH,  TIM
        INTEGER T14,  TKTHR,TIG,  TC2P, T72R, T3R
        INTEGER RGH,  RHH,  RDCF0,RDCF1,RDCF2,RDCF3
        INTEGER RDNY0,RDNY1,RDNY2,RDNY3
C
C
C MAP INTEGER SCALAR NAMES:
```

```
C
      INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG
      INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RUN
      INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO, RBTPO
      INTEGER RSNPO,DAPO, TADDC,TMFFC,RMFDC,RSNNR
      INTEGER TFCTR,RFCTR,RLCTR,RONNK,RSYNC,RBOFO
      INTEGER ADCTR,TMCTR,RMCTR,DACTR,TADPK
      INTEGER RNCOR,RERRS
C
C
C COMMON BLOCK - BUFFER AND SCALAR NAMES:
C
      COMMON /BSIDS/
     ) DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3,
     ) DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU,
     ) DVLUW,
     ) CTMC1,CTMC2,CTMC3,CTMK1,CTMK2,CTMK3,
     ) CTMK4,CTMK5,CTMK6,CTMG, CTMDG,CTMU,
     ) NISTU,
     ) TADBA,TADBB,TADBC,TSRA, TSRB, TONES,
     ) TSP0, TSP1, TSP,  THAND,TMSP, TMSPZ,
     ) TCOST,TKSP, TKSPM,TKSPI,TPSP, TPSPR,
     ) TPSP1,TRP,  TC,   TWORK,TIC,  TCH,
     ) TE10, TE11, TE1,  THAN1,TWE1, TRS,
     ) TLMU, TRSP, TKE,  TKM,  TIK,  TFILT,
     ) TAM,  TAMR, TE2,  TFAC, TAMS, TGFAC,
     ) TIDG, TWU,  TU1,  TUN,  TRNB, TRN,
     ) TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB,
     ) TMTC, TMDMA,TMDMB,TRPP,
     ) RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSRA,
     ) RSRB, RUNA, RUNB, RDGN, RCN,  REN,
     ) RUN,  RUNN, RVN,  RRNB, RRN,  RRNDA,
     ) RRNDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     ) TALPH,TPHST,TOCH, TFSZI,TMN,  TIN,
     ) T14,  TKTMR,TIG,  TC2P, T72R, T3R,
     ) NGN,  RNN,  RDCF0,RDCF1,RDCF2,RDCF3,
     ) RDMY0,RDMY1,RDMY2,RDMY3,
     ) TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG,
     ) RBTFA,RBTFB,RSNFA,RSNFB,RUN,
     ) ADPO, TSRPO,TBTPO,TMPO, RMPO, RBTPO,
     ) RSNPO,DAPO, TADDC,TMFFC,RMFDC,RSNNR,
     ) TFCTR,RFCTR,RLCTR,RONNK,RSYNC,RBOFO,
     ) ADCTR,TMCTR,RMCTR,DACTR,TADPK,
     ) RNCOR,RERRS
C
C
C COMMON BLOCK - BUFFER ADDRESSES AND SIZES
C       NEEDED FOR INIT OF NON-SWAP BUFFERS
C       AND FOR FILE-TO-FILE MODE OF OPERATION
```



```
C
      COMMON /BAS/
     ) ANISTU,SNISTU,
     ) ATADBA,STADBA,
     ) ATADBB,STADBB,
     ) ATADBC,STADBC,
     ) ATBTC, STBTC,
     ) ATMDMA,STMDMA,
     ) ATMDMB,STMDMB,
     ) ARMDMA,SRMDMA,
     ) ARMDMB,SRMDMB,
     ) ARMDMC,SRMDMC,
     ) ARSNKC,SRSNKC,
     ) ARDABA,SRDABA,
     ) ARDABB,SRDABB
C
CCCCCCC END OF BBN16-COMMON CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      DIMENSION IST(1)
C
      TYPE 10
      TYPE 20,IST(TSRFA),IST(ADPO), IST(TFCTR),           IST(RSYNC
      TYPE 30,IST(TSRFB),IST(TSRPO),IST(RFCTR),           IST(RBOFO
      TYPE 40,IST(TBTFA),IST(TBTPO),                      IST(RLCTR
      TYPE 50,IST(TBTFB),IST(TMPO), IST(ADCTR),IST(TADDC)
      TYPE 60,IST(RBTFA),IST(RMPO), IST(TMCTR),IST(TMFFC),IST(RONNK
      TYPE 70,IST(RBTFB),IST(RBTPO),IST(RMCTR),IST(RMFDC)
      TYPE 80,IST(RSNFA),IST(RSNPO),IST(DACTR),IST(RSNNR),IST(TADPK
      TYPE 90,IST(RSNFB),IST(DAPO)
C
 10   FORMAT(/,' SPEECH CODER STATE VARIABLES:',/)
 20   FORMAT(' TSRFA:',I2,'  ADPO: ',I3,'  TFCTR:',I6,'            '
     )6X,'  RSYNC:',I6)
 30   FORMAT(' TSRFB:',I2,'  TSRPO:',I3,'  RFCTR:',I6,'            '
     )6X,'  RBOFO:',I6)
 40   FORMAT(' TBTFA:',I2,'  TBTPO:',I3,'           ',6X,'            '
     )6X,'  RLCTR:',I6)
 50   FORMAT(' TBTFB:',I2,'  TMPO: ',I3,'  ADCTR:',I6,'  TADDC:'
     )I6)
 60   FORMAT(' RBTFA:',I2,'  RMPO: ',I3,'  TMCTR:',I6,'  TMFFC:'
     )I6,'  RONNK:',I6)
 70   FORMAT(' RBTFB:',I2,'  RBTPO:',I3,'  RMCTR:',I6,'  RMFDC:'
     )I6)
 80   FORMAT(' RSNFA:',I2,'  RSNPO:',I3,'  DACTR:',I6,'  RSNNR:'
     )I6,'  TADPK:',I6)
 90   FORMAT(' RSNFB:',I2,'  DAPO: ',I3)
C
      RETURN
      END
```

```
C  <DCA16>WBN16F.FOR.1    Mon 10-Nov-88 6:38PM          Page 1

C [WBND3]<DCA16>BWN16F.FOR.1, 10-Nov-88 18:38:44, Ed: KFIELD
        SUBROUTINE WBN16F
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                         WBN16F
C
C       WBN16 FUNCTION PREBINDING & FUNCTION LIST DEFINITIONS
C
C       CALLED BY WBN16R
C
C KFIELD 9/88
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C FUNCTION LIST NAMES
C
        INTEGER WBNNTS,BWNLP,WBNWFT,WBNTIM,WBNRST
        INTEGER ANLZA,ANLZB,SVNZA,SVNZB
C
C
C OTHER MAP-RELATED VARIABLES
C
        INTEGER RL,CMPLX,FXD,CNTG,LNG,SHRT
        INTEGER PRCSR,AP,WK
        INTEGER A,B,IG3,ISET,ICLR
        INTEGER WDNSCL,ADAN,ADN,IDS2,WDNSA,ADANSA,ADNSA
C
C
C MAP PREBINDING BUFFER
C
        INTEGER PBBUF
C
C
C [WBND3]<MAP>WBN16-COMMON.FOR.12, 31-Oct-88 23:17:03, Ed: WOLF
C
C
C MAP BUFFER NAMES:
C
        INTEGER DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3
        INTEGER DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU
        INTEGER DVLUF
        INTEGER CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3
        INTEGER CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU
        INTEGER WISTU
        INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, TONES
        INTEGER TSP0, TSP1, TSP,  THAN0,TWSP, TWSPZ
        INTEGER TCOST,TXSP, TXSPR,TXSPI,TPSP, TPSPR
        INTEGER TPSPI,TRP,  TC,   TWORK,TIC,  TCW
        INTEGER TE10, TE11, TE1,  THAN1,TWE1, TRS
        INTEGER TLWU, TRSP, TKE,  TKW,  TIK,  TFILT
```

```
C  <DCA16>BWN16F.FOR.1    Mon 10-Nov-88 6:38PM          Page 1:1

        INTEGER TAN,  TANR, TE2,  TFAC, TANS, TGFAC
        INTEGER TIDG, TUO,  TQ1,  TVH,  TRH0, TRH
        INTEGER TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB
        INTEGER TWTC, TWUNA,TWUNB,TRPP
        INTEGER RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSRA
        INTEGER WSNB, RUHA, RUHB, RDGH, WCH,  WKH
        INTEGER RWH,  RVHM, RVH,  RRH0, RRH,  RRHDA
        INTEGER WRHDB,WSNKA,WSNKB,WSNKC,WDABA,WDABB
C
C
C MAP REAL SCALAR NAMES:
C
        INTEGER TALPH,TPWST,TOCN, TFSZI,TMH,  TIH
        INTEGER T14,  TKTHR,TIG,  TC2P, T72R, T3R
        INTEGER RGH,  RMH,  RDCF0,RDCF1,RDCF2,RDCF3
        INTEGER RDMY0,RDMY1,RDMY2,RDMY3
C
C
C MAP INTEGER SCALAR NAMES:
C
        INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG
        INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RUN
        INTEGER ADPO, TSWPO,TWTPO,TMPO, WMPO, WBTPO
        INTEGER RSNPO,DAPO, TADDC,TMFPC,RMFDC,RSNWR
        INTEGER TFCTR,RFCTR,RLCTR,RDNWK,RSYNC,WBOFD
        INTEGER ADCTR,TMCTR,BMCTR,DACTR,TADPK
        INTEGER WNCON,BEWRS
C
C
C COMMON BLOCK - BUFFER AND SCALAR NAMES:
C
        COMMON /BSIDS/
     )  DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3,
     )  DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU,
     )  DVLUF,
     )  CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3,
     )  CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU,
     )  WISTU,
     )  TADBA,TADBB,TADBC,TSRA, TSRB, TONES,
     )  TSP0, TSP1, TSP,  THAN0,TWSP, TWSPZ,
     )  TCOST,TXSP, TXSPR,TXSPI,TPSP, TPSPR,
     )  TPSPI,TRP,  TC,   TWORK,TIC,  TCW,
     )  TE10, TE11, TE1,  THAN1,TWE1, TRS,
     )  TLWU, TRSP, TKE,  TKW,  TIK,  TFILT,
     )  TAN,  TANR, TE2,  TFAC, TANS, TGFAC,
     )  TIDG, TUO,  TQ1,  TVH,  TRH0, TRH,
     )  TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB,
     )  TWTC, TWUNA,TWUNB,TRPP,
     )  RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSRA,
     )  WSNB, RUHA, RUHB, RDGH, WCH,  RKH,
     )  RWH,  RVHM, RVH,  RRHB, RRH,  RRHDA,
```

```
     )  RRHDB,RSMKA,RSMKB,RSMKC,RDABA,RDABB,
     )  TALPH,TPHST,TDCH, TFSII,TMN,  TIN,
     )  T14,  TKTHR,TIG,  TC2P, T72R, T3R,
     )  HGM,  HMM,  RDCF0,RDCF1,RDCF2,RDCF3,
     )  RDNY0,RDNY1,RDNY2,RDNY3,
     )  TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFKNG,
     )  RNTFA,RNTFB,RSMFA,RSMFB,HUN,
     )  ADPO, TSRPO,TBTPO,TMPO, RMPO, RBTPO,
     )  RSMPO,DAPO, TADDC,TMFFC,RMFDC,RSMMR,
     )  TFCTR,RFCTR,RLCTR,RONMK,RSYNC,RDOFU,
     )  ADCTN,TMCTN,RMCTN,DACTN,TADPK,
     )  RMCOR,RERRS
C
C
C COMMON BLOCK - BUFFER ADDRESSES AND SIZES
C       NEEDED FOR INIT OF NON-SWAP BUFFERS
C       AND FOR FILE-TO-FILE MODE OF OPERATION
C
        COMMON /BAS/
     )  AHISTO,SHISTO,
     )  ATADBA,STADBA,
     )  ATADBB,STADBB,
     )  ATADBC,STADBC,
     )  ATBTC, STBTC,
     )  ATHDMA,STHDMA,
     )  ATHDMB,STHDMB,
     )  ARHDMA,SRHDMA,
     )  ARHDMB,SRHDMB,
     )  ARHDMC,SRHDMC,
     )  ARSMKC,SRSMKC,
     )  ARDABA,SRDABA,
     )  ARDABB,SRDABB
C
CCCCCCC END OF BBN16-COMMON CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C COMMON BLOCK - FUNCTION LIST NAMES
C
        COMMON /FLS/
     )  BBNRTS,BBNLP,BBNFFT,BBNTIN,BBNRST
C
C
        DATA PKCSR/1/,AP/1/,NK/1/
        DATA RL/0/,CMPLX/1/,FRD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA A,B/-2,0/,IG3,ISET,ICLR/3,1,0/
        DATA IBUS1/64/,IBUS2/128/,IBUS3/192/
C
C SCROLL IDENTIFIERS AND STARTING ADDRESSES
        DATA NOHSCL,ADAN,AON,IOS2/16,23,22,2/
        DATA NUMSA,ADANSA,AONSA/1,0,0/
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DEFINE FUNCTION LIST NAMES
C
        ANLZA = 1
        ANLZB = 2
        SYNZA = 3
        SYNZB = 4
C
C REAL TIME SYSTEM STARTUP:
        BBNRTS = 10
C
C REAL TIME SYSTEM LOOP:
        BBNLP  = 11
C
C FILE-TO-FILE SYSTEM:
        BBNFFT = 12
C
C TIMING SYSTEM:
        BBNTIN = 13
C
C RESTARTING REAL-TIME SYSTEM:
        BBNRST = 14
C
^L
```

```
C  <UCA16>BBN16F.FOR.1   Mon 10-Nov-80 6:30PM        Page 3

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DEFINE PREBINDING BUFFER, AND PREBIND NECESSARY FUNCTIONS
C
C PUT PREBINDING BUFFER AT $8000 = 32768. ON BUS1
C                 WITH SIZE $1800 =  6144.
C DON'T USE BID 63, SINCE IT IS USED BY FFTLR & FFILR.
C NOTE: FFTNR & FFINR EACH USE 2 FCB SLOTS.
C
         PBBUF =     60
        APBBUF = 32768.
        SPBBUF =  6144.
C
        CALL MPCLR(IBUS1+PBBUF,APBBUF,SPBBUF,FXD,CNTG,LNG)
C
C PUT PREBOUND FUNCTIONS IN FCBS #214 - 249
C
        CALL MPCBF(PBBUF,214)
C
C ANALYSIS:
C       #214
                CALL THIST(TSP1,TSP0,TE1,TE11,TRN,TRN0)
C       #215
                CALL DPRE(TSP0,TALPH,TSRA,TPHST,TONES)
C       #216
                CALL DPRE(TSP0,TALPH,TSRB,TPHST,TONES)
C       #217
                CALL SSUM(TDCH,TSP,TFSZI)
C       #218
                CALL VMUL(TWSP,1,TSP,TDCH,THAM0,0)
C       #219 & #220
                CALL FFTLR(TXSP,1,TWSPZ,TCOST,TXSP)
C       #221
                CALL VMSQ(TPSPR,TXSPR,TXSPI)
C       #222
                CALL VCLR(TPSPI)
C       #223 & #224
                CALL FFILR(TRP,1,TPSP,TCOST,TRP)
C       #225
                CALL PITCH(TWPP,TWN,TRP,TIN,TI4,TC2P)
C       #226
                CALL NEWL(TC,TINC,TRPP,TKTHR,TRP,TaOHK)
C       #227
                CALL STAB(TCN,TC2P,TIC,TC)
C       #228
                CALL INVPF(TE10,TWN,TSP1,TCN)
C       #229
                CALL VMUL(TWE1,1,TE10,0,THAM1,0)
C       #230
                CALL DCORZ(TNS,0,TaK1,TWK1)
C       #231
                CALL AFC(TRSP,TRS,TLNU)
```

```
C  <UCA16>BBN16F.FOR.1   Mon 10-Nov-80 6:30PM        Page 3:1

C       #232
                CALL MNLQ6(TKE,TKTHR,TKN,TRSP,TIK)
C       #233
                CALL VKTOA(TAN,TKN)
C       #234
                CALL DCORZ(TE2,0,TE1,TAHR)
C       #235
                CALL VMUL(TANS,1,TAN,0,TFAC,0)
C       #236
                CALL GAIN(TGFAC,TIG,TIDG,T72K,T22,T3K)
C       #237
                CALL APC(TIUA,TMN,TSP0,TFILT,TGFAC,TRN,TVN,Tu1)
C       #238
                CALL APC(TIUB,TMN,TSP0,TFILT,TGFAC,TRN,TVN,Tu1)
C       #239
                CALL GTHR(TSNKA,TIG,TIDG,TIN,TIC,TIK)
C       #240
                CALL GTHR(TSNKB,TIG,TIDG,TIN,TIC,TIK)
C
C SYNTHESIS:
C       #241
                CALL DEAL(RSRA,RGN,RDGN,RMN,RCN,RKN)
C       #242
                CALL DEAL(RSRB,RGN,RDGN,RMN,RCN,RKN)
C       #243
                CALL VMOV(RRN,RRN0)
C       #244
                CALL SCLRES(RWN,RGN,RDGN,RUNA)
C       #245
                CALL SCLRES(RWN,RGN,RDGN,RUNB)
C       #246
                CALL VLTSY(RVN,RVHN,RKN,RWN)
C       #247
                CALL PITSYN(RRN,RMN,RVN,RCN)
C       #248
                CALL DFL22(RRHDA,RDCF0,RRH0,RDMY0)
C       #249
                CALL DFL22(RRHDB,RDCF0,RRH0,RDMY0)
C
C
        CALL MPTBF(0)
^L
```

```
C  <DCA16>BBN16F.FOR.1   Mon 10-Nov-88 6:30PM        Page 4

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C DEFINE UPPER LEVEL SYSTEM FUNCTION LISTS:
C  BBNRTS, BBNLP, BBNFFT, BBNTIN, BBNRST
C
C
C BBNRTS: REAL TIME SYSTEM STARTUP FUNCTION LIST.
C
        CALL MPBFL(BBNRTS)
                CALL MPRMS(MDMSCL,IDS2,MDMSA)
                CALL MPRMS(ADAM,IDS2,ADAMSA)
                CALL MPRMS(AUM, IDS2,AUMSA)
                CALL MPIST(RUN,1)
                CALL MPINL(RUN,NE,0,BBNLP)
        CALL MPEFL(BBNRTS)
^L
```

```
C  <DCA16>BBN16F.FOR.1   Mon 10-Nov-88 6:30PM        Page 5

C
C BBNLP: REAL TIME SYSTEM LOOP FUNCTION LIST.
C
        CALL MPBFL(BBNLP)
                CALL MPIFF(TSNFA,TBTFB,ANLZA)
                CALL MPIFF(RBTFB,RSNFA,SYNZA)
                CALL MPIFF(TSRFB,TBTFA,ANLZB)
                CALL MPIFF(RBTFA,RSNFB,SYNZB)
        CALL MPEFL(BBNLP)
^L
```

```
C
C BBNRST: REAL TIME SYSTEM RESTART FUNCTION LIST.
C
      CALL NPBFL(BBNRST)
            CALL NPIST(RUN,1)
            CALL NPINL(RUN,NE,N,BBNLP)
      CALL NPEFL(BBNRST)
^L
```



```
C
C BBNFFT: FILE-TO-FILE SYSTEM FUNCTION LIST.
C
      CALL NPBFL(BBNFFT)
            CALL NPNT(PRCSR,AP)
            CALL ADINT
            CALL RMINT
            CALL NPIFF(TSRFA,TBTFB,ANLZA)
            CALL NPIFF(RBTFB,RSNFA,SYNZA)
            CALL NPIFF(TSRFB,TBTFA,ANLZB)
            CALL NPIFF(RBTFA,RSNFB,SYNZB)
            CALL NPXT(PRCSR,AP)
            CALL TMINT
            CALL DAINT
      CALL NPEFL(BBNFFT)
^L
```

```
C
C BBNTIN: TIMING SYSTEM FUNCTION LIST.
C       NOTE: THIS FUNCTION LIST INCLUDES ONLY THE
C       SIGNAL PROCESSING PORTIONS OF THE SYSTEM.
C       IT DOES NOT REPRESENT A FUNCTIONING SPEECH CODING
C       SYSTEM.  THE FOLLOWING ROUTINES ARE NOT INCLUDED:
C               ADINT, THINT, RHINT, DAINT.
C
        CALL MPBFL(BBNTIN)
                CALL MPXFL(ANLZA)
                CALL MPXFL(SYNZA)
        CALL MPEFL(BBNTIN)
^L
```



```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C DEFINE LOWER LEVEL FUNCTION LISTS:
C  ANLZA, ANLZB, SYNZA, SYNZB
C
C
C ANLZA: ANALYSIS FROM TSRA -> TSRKA;
C          PRTECT FROM TSRKB -> TBTB.  (PROPAR & PRORES)
C
        CALL MPBFL(ANLZA)
C
C SET G3 FOR SCOPE SYNC AT START OF TRANSMITTER
                CALL MPGSC(IG3,ISET)
C
C MONITOR PEAK INPUT LEVEL
                CALL ADPEAK(TSRA)
C
C TRANSMITTER FRAME HISTORY BUFFER UPDATES...
C*      CALL THIST(TSP1,TSP0,TE1,TE11,TRH,TRH0)
                CALL MPXBF(214)
C
C PREEMPHASIS...
C*      CALL DPRE(TSP0,TALPH,TSRA,TPHST,TONES)
                CALL MPXBF(215)
C
C DONE WITH TSRA...
                CALL MPWT(PRCSR,AP)
                CALL MPIST(TSRFA,0)
C
C PITCH FILTER ANALYSIS...
C*      CALL SSUM(TDCN,TSP,TFSZI)
                CALL MPXBF(217)
C*      CALL VMUL(TWSP,1,TSP,TDCN,THAM0,0)
                CALL MPXBF(218)
C*      CALL FFTLR(TXSP,1,TWSP2,TCOST,TXSP)
                CALL MPXBF(219)
                CALL MPXBF(220)
C*      CALL VMSW(TPSPW,TXSPW,TXSPI)
                CALL MPXBF(221)
C*      CALL VCLR(TPSPI)
                CALL MPXBF(222)
C*      CALL FFILR(TRP,1,TPSP,TCOST,TRP)
                CALL MPXBF(223)
                CALL MPXBF(224)
C*      CALL PITCH(TRPP,TMH,TRP,TIN,T14,TC2P)
                CALL MPXBF(225)
C*      CALL NEWL(TC,TINC,TRPP,TKTHR,TRP,TWORK)
                CALL MPXBF(226)
C*     -CALL STAB(TCH,TC2P,TIC,TC)
                CALL MPXBF(227)
C
C INVERSE PITCH FILTER...
```

```
C*      CALL INVPF(TE10,TMH,TSP1,TCH)
              CALL MPXBF(228)
C
C SPECTRAL FILTER ANALYSIS...
C*      CALL VMUL(TWE1,1,TE10,0,THAM1,0)
              CALL MPXBF(229)
C*      CALL DCORZ(TRS,0,TWE1,TWE1)
              CALL MPXBF(230)
C*      CALL NFC(TRSP,TRS,TLNU)
              CALL MPXBF(231)
C*      CALL NMLQ6(TKE,TKTHR,TKH,TRSP,TIK)
              CALL MPXBF(232)
C
C INVERSE SPECTRAL FILTER...
C*      CALL VKTOA(TAH,TKH)
              CALL MPXBF(233)
C*      CALL DCORZ(TE2,0,TE1,TAHR)
              CALL MPXBF(234)
C
C ERROR PROTECTION ON PREVIOUS FRAME PARAMETER DATA
C (HIDDEN UNDER DCORZ FUNCTION)...
              CALL PROPAR(0)
C
C NOISE SHAPING FILTER CALCULATION...
C*      CALL VMUL(TANS,1,TAH,0,TFAC,0)
              CALL MPXBF(235)
C
C GAIN FACTOR CALCULATION...
C*      CALL GAIN(TGFAC,TIG,TIDG,T72R,TE2,T3R)
              CALL MPXBF(236)
C
C APC RESIDUAL CALCULATION ("TIUA" IS EQUIVALENT TO START OF "TSNKA")...
C*      CALL APC(TIUA,TMH,TSP0,TFILT,TGFAC,TRH,TVH,TQI)
              CALL MPXBF(237)
C
C ERROR PROTECTION ON PREVIOUS FRAME RESIDUAL DATA
C (HIDDEN UNDER APC FUNCTION)...
              CALL PRORES(0)
              CALL NPIST(TBTFB,1)
C
C TRANSMITTER DATA COLLECTION...
C*      CALL GTHR(TSNKA,TIG,TIDG,TIN,TIC,TIK)
              CALL MPXBF(239)
C
        CALL MPEFL(ANLZA)
^L
```



```
C
C ANLZB: ANALYSIS FROM TSRB -> TSNKB;
C        PRTECT FROM TSNKA -> TBTA.  (PROPAR & PRORES)
C
        CALL MPBFL(ANLZB)
C
C SET G3 FOR SCOPE SYNC AT START OF TRANSMITTER
              CALL MPGSC(IG3,ISET)
C
C MONITOR PEAK INPUT LEVEL
              CALL ADPEAK(TSRB)
C
C TRANSMITTER FRAME HISTORY BUFFER UPDATES...
C*      CALL THIST(TSP1,TSP0,TE1,TE11,TRH,TRH0)
              CALL MPXBF(214)
C
C PREEMPHASIS...
C*      CALL DPRE(TSP0,TALPH,TSRB,TPHST,TONES)
              CALL MPXBF(216)
C
C DONE WITH TSRB...
              CALL MPWT(PRCSR,AP)
              CALL NPIST(TSRFB,0)
C
C PITCH FILTER ANALYSIS...
C*      CALL SSUM(TDCH,TSP,TFSZI)
              CALL MPXBF(217)
C*      CALL VMUL(TWSP,1,TSP,TDCH,THAM0,0)
              CALL MPXBF(218)
C*      CALL FFTLR(TXSP,1,TWSPZ,TCOST,TXSP)
              CALL MPXBF(219)
              CALL MPXBF(220)
C*      CALL VMSQ(TPSPR,TXSPR,TXSPI)
              CALL MPXBF(221)
C*      CALL VCLR(TPSPI)
              CALL MPXBF(222)
C*      CALL FFILR(TRP,1,TPSP,TCOST,TRP)
              CALL MPXBF(223)
              CALL MPXBF(224)
C*      CALL PITCH(TRPP,TMH,TRP,TIM,TI4,TC2P)
              CALL MPXBF(225)
C*      CALL NEWL(TC,TINC,TRPP,TKTHR,TRP,TWORK)
              CALL MPXBF(226)
C*      CALL STAB(TCH,TC2P,TIC,TC)
              CALL MPXBF(227)
C
C INVERSE PITCH FILTER...
C*      CALL INVPF(TE10,TMH,TSP1,TCH)
              CALL MPXBF(228)
C
C SPECTRAL FILTER ANALYSIS...
```

```
C*      CALL VMUL(TWK1,1,TK18,8,THAM1,8)
                CALL MPXBF(229)
C*      CALL DCORZ(TRS,8,TWK1,TWK1)
                CALL MPXBF(230)
C*      CALL MPC(TRSP,TRS,TLMU)
                CALL MPXBF(231)
C*      CALL MWLOG(TKE,TKTHR,TKH,TRSP,TIK)
                CALL MPXBF(232)
C
C INVERSE SPECTRAL FILTER...
C*      CALL VATOA(TAH,TKH)
                CALL MPXBF(233)
C*      CALL DCORZ(TE2,8,TE1,TAHR)
                CALL MPXBF(234)
C
C ERROR PROTECTION ON PREVIOUS FRAME PARAMETER DATA
C (HIDDEN UNDER DCORZ FUNCTION)...
                CALL PROPAR(A)
C
C NOISE SHAPING FILTER CALCULATION...
C*      CALL VMUL(TAHS,1,TAH,8,TFAC,8)
                CALL MPXBF(235)
C
C GAIN FACTOR CALCULATION...
C*      CALL GAIN(TGFAC,TIG,TIDG,T72R,TE2,T3R)
                CALL MPXBF(236)
C
C APC RESIDUAL CALCULATION ('TIUB' IS EQUIVALENT TO START OF 'TSNKB')...
C*      CALL APC(TIUB,THH,TSP8,TFILT,TGFAC,TRH,TVH,TQ1)
                CALL MPXBF(238)
C
C ERROR PROTECTION ON PREVIOUS FRAME RESIDUAL DATA
C (HIDDEN UNDER APC FUNCTION)...
                CALL PRORES(A)
                CALL MPIST(TNTFA,1)
C
C TRANSMITTER DATA COLLECTION...
C*      CALL GTHR(TSNKB,TIG,TIDG,TIN,TIC,TIK)
                CALL MPXBF(240)
C
        CALL MPEFL(ANLZB)
^L
```



```
C
C SYNZA: SYNTHESIS FROM RSRA -> RSNKA;
C          DECODB/CORPAR FROM RBTB -> RSNB.
C
        CALL MPBFL(SYNZA)
C
                CALL MPGSC(IG3,ICLR)
C
C RECEIVER DATA DISTRIBUTION...
C*      CALL DEAL(RSRA,RGH,RDGH,RNH,RCH,RKH)
                CALL MPXBF(241)
C
C RECEIVER FRAME HISTORY BUFFER UPDATE...
C*      CALL VMOV(RRH,RRHB)
                CALL MPXBF(243)
C
C RESIDUAL SCALING ('RUHA' IS EQUIVALENT TO START OF 'RSRA')...
C*      CALL SCLRES(RWH,RGH,RDGH,RUHA)
                CALL MPXBF(244)
C
C SPECTRAL SYNTHESIS...
C*      CALL VLTSY(RVH,RVHM,RKH,RWH)
                CALL MPXBF(246)
C
C DECODING OF PREVIOUS FRAME RECEIVER RESIDUAL DATA &
C ERROR CORRECTION ON PREVIOUS FRAME RECEIVER PARAMETER DATA
C (PARTIALLY HIDDEN UNDER VLTSY FUNCTION)...
                CALL DECODB
                CALL CORPAR(B)
                CALL MPWT(PRCSR,AP)
                CALL MPIST(RBTFB,8)
C
C PITCH SYNTHESIS...
C*      CALL PITSYN(RRH,RNH,RVH,RCH)
                CALL MPXBF(247)
C
C DEEMPHASIS ('RRHDA' IS EQUIVALENT TO 'RSNKA')...
C*      CALL DFL22(RRHDA,RDCFB,RRHB,RDMYB)
                CALL MPXBF(248)
C
C RSNKA IS FULL...
                CALL MPWT(PRCSR,AP)
                CALL MPIST(RSNFA,1)
C
        CALL MPEFL(SYNZA)
^L
```

```
C  <DCA16>BBN16F.FOR.1   Mon 18-Nov-85 6:38PM        Page 12
C
C SYNZB: SYNTHESIS FROM RSRB -> RSNKB)
C           DECODA/CORPAR FROM RRFA -> RSRA.
C
        CALL MPBFL(SYNZB)
C
                CALL MPGSC(IG3,ICLR)
C
C RECEIVER DATA DISTRIBUTION...
C*      CALL DEAL(RSRB,RGH,RDGH,RNH,RCH,RKH)
                CALL MPXBF(242)
C
C RECEIVER FRAME HISTORY BUFFER UPDATE...
C*      CALL VHOV(RRH,RRHB)
                CALL MPXBF(243)
C
C RESIDUAL SCALING ('RUHB' IS EQUIVALENT TO START OF 'RSRB')...
C*      CALL SCLRES(RUH,RGH,RDGH,RUHB)
                CALL MPXBF(245)
C
C SPECTRAL SYNTHESIS...
C*      CALL VLTSY(RVH,RVHH,RKH,RNH)
                CALL MPXBF(246)
C
C DECODING OF PREVIOUS FRAME RECEIVER RESIDUAL DATA &
C ERROR CORRECTION ON PREVIOUS FRAME RECEIVER PARAMETER DATA
C (PARTIALLY HIDDEN UNDER VLTSY FUNCTION)...
                CALL DECODA
                CALL CORPAR(A)
                CALL MPWT(PRCSB,AP)
                CALL MPIST(RRTFA,8)
C
C PITCH SYNTHESIS...
C*      CALL PITSYN(RRH,RNH,RVH,RCH)
                CALL MPXBF(247)
C
C DEEMPHASIS ('RRHDB' IS EQUIVALENT TO 'RSNKB')...
C*      CALL DFL22(RRHDB,RDCF0,RRH0,RDHY0)
                CALL MPXBF(249)
C
C RSNKB IS FULL...
                CALL MPWT(PRCSB,AP)
                CALL MPIST(RSNFB,1)
C
        CALL MPEFL(SYNZB)
```

```
C  <DCA16>BBN16F.FOR.1   Mon 18-Nov-85 6:38PM        Page 13
C
        RETURN
        END
```

```
C  <DCA16>BBN16N.FOR.1   Mon 18-Nov-88 6:40PM         Page 1

C [BBN]<DCA16>BBN16N.FOR.1, 18-Nov-88 18:48:30, Ed: KFIELD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C THIS FILE CONTAINS MAP-300 SNAPII HOST SUPPORT FUNCTIONS
C USED BY THE BBN DCA16 SYSTEM.
C FUNCTIONS ARE ORDERED ALPHABETICALLY.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
^L

C  <DCA16>BBN16N.FOR.1   Mon 18-Nov-88 6:40PM         Page 2

C
C
C
        INTEGER FUNCTION ADINT(I)
C
C FCB # 124
        ADINT=INTFN(124,0)
        RETURN
        END
C
C
C
        INTEGER FUNCTION ADPEAK(U)
C
C USAGE: IER = ADPEAK(U)
C
        INTEGER U,FCBGN
        ADPEAK = FCBGN(102,0,0,U,0,0,0,0,0,10)
        RETURN
        END
C
C
C
        INTEGER FUNCTION APC(Y,A,U,V,W,R,S,T)
C
C USAGE:          IER = APC(Y,A,U,V,W,R,S,T)
C
C NOTE: Y,U,V,W,R,S,T ARE BUFFERS; A IS SCALAR
C       FUNCTION REQUIRES SPECIAL BINDING
C
        INTEGER Y,A,U,V,W,R,S,T,FCBGN
        APC = FCBGN(-199,Y,A,U,V,W,R,S,T,14)
        RETURN
        END
C
C
C
        INTEGER FUNCTION CORPAR(IAB)
C
C IAB=-2  => CONECT WBTA INTO WSRA
C IAB=0   => CONECT RBTB INTO WSRB
C  FCB #123
C
        CORPAR=INTFN(123,IAB)
        RETURN
        END
C
C
C
        INTEGER FUNCTION DAINT(I)
C
```

```
C  <DCA16>BBN16N.FOR.1   Mon 18-Nov-80 6:40PM          Page 2:1

C PCB @ 127
        BAINT=INTFN(127,0)
        RETURN
        END
C
C
C
        INTEGER FUNCTION DCORZ(V,IA,U,V)
C
C USAGE: IER = DCORZ(V,IA,U,V)
C
        INTEGER V,U,V,FCBGN
        IF (IA .GT. 127) CALL NPERN(-2)
        DCORZ = FCBGN(-213,V,IA,U,0,V,0,0,0,9)
        RETURN
        END
C
C
C
        INTEGER FUNCTION DEAL(V,A,U,B,V,W)
C
C USAGE: IER = DEAL(V,A,U,B,V,W)
C
        INTEGER V,A,U,B,V,W,FCBGN
        DEAL = FCBGN(212,V,A,U,B,V,0,W,0,14)
        RETURN
        END
C
C
C
        INTEGER FUNCTION DECODA
C
C USAGE: IER = DECODA
C DOES DECODE OF RESIDUAL FROM 'RBTA' TO 'RSRA' BUFFERS.
C BUFFERS ARE HARDWIRED TO APS CODE BECAUSE OF SHORTAGE OF BIDS.
C
        INTEGER FCBGN
        DECODA = FCBGN(188,0,0,0,0,0,0,0,0,1)
        RETURN
        END
C
C
C
        INTEGER FUNCTION DECODB
C
C USAGE: IER = DECODB
C DOES DECODE OF RESIDUAL FROM 'RBTB' TO 'RSRB' BUFFERS.
C BUFFERS ARE HARDWIRED TO APS CODE BECAUSE OF SHORTAGE OF BIDS.
C
        INTEGER FCBGN
        DECODB = FCBGN(189,0,0,0,0,0,0,0,0,1)
        RETURN
        END
```

```
C  <DCA16>BBN16N.FOR.1   Mon 18-Nov-80 6:40PM          Page 2:2
C
C
C
      INTEGER FUNCTION FFILR (V,N,U,V,W)
C
C          FFTIN FOR REAL DATA.
C          COMBINES FFTIN AND FEOI
C               MODIFIED TO COMBINE FEOIL AND FFTIN
C               THIS REQUIRES (2**N)+1 COMPLEX INPUT POINTS,
C               RATHER THAN (2**N) COMPLEX INPUT POINTS.
C               THE (2**N)+1ST REAL INPUT POINT IS INPUT AS SUCH,
C               INSTEAD OF IN PLACE OF THE 1ST IMAGINARY INPUT POINT
C               (1ST IMAGINARY INPUT POINT IS ALWAYS ZERO.).
C
C
C
      INTEGER FCBGN, V, U, V, W
C
      FFILR = NPEVN (63, V)
      IF (FFILR .NE. 0) RETURN
C
        J = W
      IF (W .EQ. V) J = 63
C
C
C DO FEOIL...
C
      FFILR = FCBGN (-198,U,0,U,0,0,0,U,0,18)
      IF (FFILR .NE. 0) RETURN
C
C
C DO FFTIN...
C
      FFILR = FCBGN (-206,63,N,U,0,V,0,J,0,17)
C
      RETURN
      END
C
C
C
      INTEGER FUNCTION FFTLR (V,N,U,V,W)
C
C          FFTN FOR REAL DATA.
C          COMBINES FFTN AND FEOF
C               MODIFIED TO COMBINE FFTN AND FEOFL
C               THIS RESULTS IN (2**N)+1 COMPLEX OUTPUT POINTS,
C               RATHER THAN (2**N) COMPLEX OUTPUT POINTS.
C               THE (2**N)+1ST REAL OUTPUT POINT IS OUTPUT AS SUCH,
C               INSTEAD OF IN PLACE OF THE 1ST IMAGINARY OUTPUT POINT
C               (1ST IMAGINARY OUTPUT POINT IS ALWAYS ZERO.).
C
```

```
C  <UCA10>BBN16N.FOR.1    Mon 10-Nov-80 6:40PM          Page 2:3
C
      INTEGER FCBGN, Y, U, V, W
C
      FFTLR = NPERR (63, W)
      IF (FFTLR .NE. 0) RETURN
C
C
C DO FFTN...
C
      FFTLR = FCBGN (-204,Y,N,63,0,V,0,W,0,17)
      IF (FFTLR .NE. 0) RETURN
C
C
C DO FEOFL...
C
      FFTLR = FCBGN (-197,Y,0,Y,0,0,0,V,0,10)
C
      RETURN
      END
C
C
C
        INTEGER FUNCTION GAIN(Y,A,U,B,V,C)
C
C  MUST SUPPORT FOR GAIN
C  USAGE: IERR=GAIN(Y,A,U,B,V,C)
C
        INTEGER Y,A,U,B,V,C,FCBGN
        GAIN=FCBGN(196,Y,A,U,B,V,C,0,0,3)
        RETURN
        END
C
C
C
        INTEGER FUNCTION GTHR(Y,A,U,B,V,W)
C
C MUST SUPPORT FOR GTHR
C USAGE: IERR = GTHR(Y,A,U,B,V,W)
C
        INTEGER Y,A,U,V,W,FCBGN
        GTHR = FCBGN(208,Y,A,U,W,V,0,W,0,14)
        RETURN
        END
C
C
C
        INTEGER FUNCTION MFC(Y,U,V)
C
C  MUST SUPPORT FOR MFC
C  USAGE: IERR=MFC(Y,U,V)
```

```
C  <UCA10>BBN16N.FOR.1    Mon 10-Nov-80 6:40PM          Page 2:4
C
      INTEGER Y,U,V,FCBGN
      MFC=FCBGN(198,Y,0,U,0,V,0,0,0,0)
      RETURN
      END
C
C
C
      INTEGER FUNCTION INTFN(IFCBN,IARG)
C
C CALLER FOR ADINT, TMINT, RMINT, DAINT
C  ALSO CALLER FOR PRORES, PROPAR, CORECT
C
      INTEGER FCBRG,FCBSZ,HWS,FCB,RUNMP,HRI,FLID
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),HWS,FCB(6),MPDCB(4),HRI,LE
      INTFN=0
      DO 10 I=1,11
10      FCBRG(I)=0
      FCBRG(2)=IFCBN
C
      IF(IARG.NE.0 .AND. IARG.NE.-2) INTFN=-1
      IF(INTFN.NE.0) CALL NPERR(INTFN)
      IF(INTFN.NE.0) RETURN
C
      FCBRG(4)=IARG
      INTFN=RUNMP(FCBRG(1),FCBSZ(1,2))
      RETURN
      END
C
C
C
      INTEGER FUNCTION INVPF(Y,A,U,V)
C
C  MUST SUPPORT FOR INVPF
C  USAGE: IERR=INVPF(Y,A,U,V)
C
      INTEGER Y,A,U,V,FCBGEN
      INVPF=FCBGN(167,Y,A,U,0,V,0,0,0,9)
      RETURN
      END
C
C
C
      INTEGER FUNCTION NEWL(Y,IA,U,B,V,W)
C
C USAGE: IER = NEWL(Y,IA,U,B,V,W)
C
      INTEGER Y,U,B,V,W,FCBGN
      IF (IA .GT. 127) CALL NPERR(-2)
      NEWL = FCBGN(-142,Y,IA,U,B,V,0,W,0,14)
      RETURN
      END
```

```
C  <UCA16>RUNION.FOR.1   Mon 18-Nov-85 6:40PM          Page 215

C
C
C
      INTEGER FUNCTION MPGSC(GFLAG,SETCLR)
C
C SUPPORT MODULE FOR MPGSC -- GFLAG SET/CLEAR
C
C CALLING SEQUENCE:
C  IER = MPGSC(GFLAG,SETCLR)
C  WHERE:
C   GFLAG = G-FLAG NUMBER (0,1,2,3)
C   SETCLR = 0 TO CLEAR IT, 1 TO SET IT
C
      INTEGER GFLAG,SETCLR
      INTEGER FCBRG,FCBSZ,NWS,FCB,RUNMP,HRI,FLID
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),NWS,FCB(6),MPDCB(4),HRI,LEVEL
C
      MPGSC = 0
C
      DO 10 I=1,11
   10 FCBRG(I) = 0
C
C FCB 0106
C
      FCBRG(2) = 106
C
      IF ((SETCLR .LT. 0) .OR. (SETCLR .GT. 1)) MPGSC = -2
      IF ((GFLAG .LT. 0) .OR. (GFLAG .GT. 3)) MPGSC = -1
C
      IF (MPGSC .NE. 0) CALL MPERR(MPGSC)
      IF (MPGSC .NE. 0) RETURN
C
      FCBRG(5) = GFLAG
      FCBRG(7) = SETCLR
C
      MPGSC = RUNMP(FCBRG(1),FCBSZ(1,5))
C
      RETURN
      END
C
C
C
      INTEGER FUNCTION MPIFF(ISA,ISB,FLID)
C
C SUPPORT MODULE FOR THE INTEGER "IF A.NE.0 .AND. B.EQ.0"
C  CONDITIONAL FCB.
C
C CALLING SEQUENCE:
C  IER = MPIFF(ISA,ISB,FLID)
C  WHERE:
C   ISA = INTEGER SCALAR ID (0 .LE. ISA .LE. 127)
```

```
C  <UCA16>RUNION.FOR.1   Mon 18-Nov-85 6:40PM          Page 216

C   ISB = INTEGER SCALAR ID (0 .LE. ISB .LE. 127)
C   FLID = FUNCTION LIST ID (0 .LE. FLID .LE. 63)
C
      INTEGER FCBRG,FCBSZ,NWS,FCB,RUNMP,HRI,FLID
      COMMON/MPZZZ/FCBRG(11),FCBSZ(11,7),NWS,FCB(6),MPDCB(4),HRI,LEVE
C
      MPIFF = 0
C
      DO 10 I=1,11
   10 FCBRG(I) = 0
C
C FCB 0105
C
      FCBRG(2) = 105
C
      IF ((FLID .LT. 0) .OR. (FLID .GT. 63)) MPIFF = -3
      IF ((ISB .LT. 0) .OR. (ISB .GT. 127)) MPIFF = -2
      IF ((ISA .LT. 0) .OR. (ISA .GT. 127)) MPIFF = -1
C
      IF (MPIFF .NE. 0) CALL MPERR(MPIFF)
      IF (MPIFF .NE. 0) RETURN
C
      FCBRG(3) = ISA
      FCBRG(5) = ISB
      FCBRG(7) = FLID
C
      MPIFF = RUNMP(FCBRG(1),FCBSZ(1,5))
C
      RETURN
      END
C
C
C
      INTEGER FUNCTION NWLOG(Y,A,U,V,W)
C
C  HOST SUPPORT FOR NWLOG
C  USAGE: IERR=NWLOG(Y,A,U,V,W)
C
      INTEGER Y,A,U,V,W,FCBGN
C
      NWLOG=FCBGN(135,Y,A,U,0,V,0,W,0,17)
      RETURN
      END
C
C
C
      INTEGER FUNCTION PITCH(Y,A,U,B,C,D)
C
C  HOST SUPPORT FOR PITCH
C  USAGE: IERR=PITCH(Y,A,U,B,C,D)
```

```
C
      INTEGER V,A,U,B,C,D,FCBGN
      IBUF=V
      PITCH=FCBGN(134,V,A,U,B,IBUF,C,IBUF,D,14)
      RETURN
      END
C
C
C
      INTEGER FUNCTION PITSYN(V,A,U,V)
C
C  HOST SUPPORT FOR PITSYN
C  USAGE: IERR=PITSYN(V,A,U,V)
C
      INTEGER V,A,U,V,FCBGN
      PITSYN=FCBGN(203,V,A,U,0,V,0,0,0,9)
      RETURN
      END
C
C
C
      INTEGER FUNCTION PROPAR(IAB)
C
C IAB=-2 => PROTECT (PARAMETERS) TSNKA INTO TBTA
C IAB=0  => PROTECT (PARAMETERS) TSNKB INTO TBTB
C  FCB #122
C
      PROPAR=INTFN(122,IAB)
      RETURN
      END
C
C
C
      INTEGER FUNCTION PRORES(IAB)
C
C IAB=-2 => PROTECT (RESIDUAL) TSNKA INTO TBTA
C IAB=0  => PROTECT (RESIDUAL) TSNKB INTO TBTB
C  FCB #121
C
      PRORES=INTFN(121,IAB)
      RETURN
      END
C
C
C
      INTEGER FUNCTION RNINT(I)
C
C FCB # 126
      RNINT=INTFN(126,0)
      RETURN
      END
```



```
C
C
C
      INTEGER FUNCTION SCLRES(V,A,U,V)
C
C  HOST SUPPORT FOR SCLRES
C  USAGE: IERR=SCLRES(V,A,U,V)
C
      INTEGER V,A,U,V,FCBGN
      SCLRES=FCBGN(202,V,A,U,0,V,0,0,0,9)
      RETURN
      END
C
C
C
      INTEGER FUNCTION STAB(V,A,U,V)
C
C  HOST SUPPORT FOR STAB
C  USAGE: IERR=STAB(V,A,U,V)
C
      INTEGER V,A,U,V,FCBGN
      STAB=FCBGN(150,V,A,U,0,V,0,0,0,9)
      RETURN
      END
C
C
C
      INTEGER FUNCTION THIST(V,U,V,W,R,S)
C
C USAGE: IER = THIST(V,U,V,W,R,S)
C
C NOTE: ALL ARGS ARE BUFFERS: SPECIAL BINDING IS REQ'D.
C
      INTEGER V,U,V,W,R,S,FCBGN
      THIST = FCBGN(-201,V,U,V,W,R,S,0,0,3)
      RETURN
      END
C
C
C
      INTEGER FUNCTION TMINT(I)
C
C FCB # 125
      TMINT=INTFN(125,0)
      RETURN
      END
C
C
C
      INTEGER FUNCTION VKTOA(V,B)
C
```

```
C  MUST SUPPORT FOR VKTOA
C  USAGE: IERR=VKTOA(V,U)
C
       INTEGER V,U,FCBGN
C
       VKTOA=FCBGN(133,V,0,U,0,0,0,0,0,11)
       RETURN
       END
C
C
C
       INTEGER FUNCTION VLTSV(V,U,V,W)
C
C USAGE  IER = VLTSV(V,U,V,W)
C
C  NOTE: USE CALLING SEQUENCE 017 (V,A,U,V,W), WITH A=0
C
       INTEGER FCBGN,V,U,V,W
       VLTSV = FCBGN(132,V,0,U,0,V,0,W,0,17)
       RETURN
       END
```

```
C (BBNB)<UCA16>BBN16I.FOR.1, 16-Nov-88 18:41:28, Ed: KFIELD
      SUBROUTINE BBN16I
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                      BBN16I
C
C BBN16 MAP-300 BUFFER & SCALAR INITIALIZATION
C
C CALLED BY BBN16R
C
C KFIELD 7/88
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C HOST ARRAYS FOR INITIALIZATION
C
      REAL HWORK(265)
      INTEGER IWORK(522)
C
C
C MAP-RELATED VARIABLES
C
      INTEGER NL,CMPLX,FXD,CHTG,LNG,SHRT
      INTEGER PRCSR,AP
C
C
C (BBNB)<MAP>BBN16-COMMON.FOR.12, 31-Oct-88 23:17:03, Ed: WOLF
C
C
C MAP BUFFER NAMES:
C
      INTEGER DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3
      INTEGER DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU
      INTEGER DVLUF
      INTEGER CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3
      INTEGER CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU
      INTEGER HISTO
      INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, TONES
      INTEGER TSPB, TSP1, TSP,  THAMB,TWSP, TWSPZ
      INTEGER TCOST,TXSP, TXSPR,TXSPI,TPSP, TPSPR
      INTEGER TPSPI,TRP,  TC,   TWORK,TIC,  TCH
      INTEGER TE1B, TE11, TE1,  THAM1,TWE1, TRS
      INTEGER TLMU, TRSP, TKE,  TKH,  TIK,  TFILT
      INTEGER TAH,  TAHR, TE2,  TFAC, TANS, TGFAC
      INTEGER TIDG, TWD,  TW1,  TVH,  TRHB, TRH
      INTEGER TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB
      INTEGER TBTC, THDHA,THDHB,TRPP
      INTEGER RHDHA,RHDHB,RHDHC,RBTA, RBTB, RSRA
      INTEGER RSRB, RUHA, RUHB, RDGH, RCH,  RKH
      INTEGER RWH,  RVHH, RVH,  RRHB, RRH,  RRHDA
      INTEGER RRHDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB
```



```
C
C
C MAP REAL SCALAR NAMES:
C
      INTEGER TALPH,TPHST,TDCH, TFSZ1,THH,  TIH
      INTEGER T14,  TKTHR,TIG,  TC2P, T72R, T3H
      INTEGER RGH,  RHH,  RDCF0,RDCF1,RDCF2,RDCF3
      INTEGER RDMY0,RDMY1,RDMY2,RDMY3
C
C
C MAP INTEGER SCALAR NAMES:
C
      INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG
      INTEGER RBTFA,RBTFB,RSNFA,RSNFB,RUN
      INTEGER ADPO, TSRPO,TBTPO,THPO, RHPO, RBTPO
      INTEGER RSNPO,DAPO, TADDC,THFPC,RHFDC,RSNNR
      INTEGER TFCTR,RFCTR,RLCTR,RUNHK,RSYNC,RBUFU
      INTEGER ADCTR,THCTR,RHCTR,DACTR,TADPK
      INTEGER RHCOR,RERRS
C
C
C COMMON BLOCK - BUFFER AND SCALAR NAMES:
C
      COMMON /BSIDS/
     $ DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3,
     $ DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU,
     $ DVLUF,
     $ CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3,
     $ CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU,
     $ HISTO,
     $ TADBA,TADBB,TADBC,TSRA, TSRB, TONES,
     $ TSPB, TSP1, TSP,  THAMB,TWSP, TWSPZ,
     $ TCOST,TXSP, TXSPR,TXSPI,TPSP, TPSPR,
     $ TPSPI,TRP,  TC,   TWORK,TIC,  TCH,
     $ TE1B, TE11, TE1,  THAM1,TWE1, TRS,
     $ TLMU, TRSP, TKE,  TKH,  TIK,  TFILT,
     $ TAH,  TAHR, TE2,  TFAC, TANS, TGFAC,
     $ TIDG, TWD,  TW1,  TVH,  TRHB, TRH,
     $ TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB,
     $ TBTC, THDHA,THDHB,TRPP,
     $ RHDHA,RHDHB,RHDHC,RBTA, RBTB, RSRA,
     $ RSRB, RUHA, RUHB, RDGH, RCH,  RKH,
     $ RWH,  RVHH, RVH,  RRHB, RRH,  RRHDA,
     $ RRHDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     $ TALPH,TPHST,TDCH, TFSZ1,THH,  TIH,
     $ T14,  TKTHR,TIG,  TC2P, T72R, T3H,
     $ RGH,  RHH,  RDCF0,RDCF1,RDCF2,RDCF3,
     $ RDMY0,RDMY1,RDMY2,RDMY3,
     $ TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG,
     $ RBTFA,RBTFB,RSNFA,RSNFB,RUN,
     $ ADPO, TSRPO,TBTPO,THPO, RHPO, RBTPO,
     $ RSNPO,DAPO, TADDC,THFPC,RHFDC,RSNNR,
     $ TFCTR,RFCTR,RLCTR,RUNHK,RSYNC,RBUFU,
     $ ADCTR,THCTR,RHCTR,DACTR,TADPK,
     $ RHCOR,RERRS
```

```
C
C
C COMMON BLOCK - BUFFER ADDRESSES AND SIZES
C       NEEDED FOR INIT OF NON-SNAP BUFFERS
C       AND FOR FILE-TO-FILE MODE OF OPERATION
C
        COMMON /BAS/
     $  ANISTO,SNISTO,
     $  ATADAA,STADAA,
     $  ATADBB,STADBB,
     $  ATADBC,STADBC,
     $  ATBTC, STBTC,
     $  ATMDMA,STMDMA,
     $  ATMDMB,STMDMB,
     $  ARNDMA,SRNDMA,
     $  ARNDMB,SRNDMB,
     $  ANNDMC,SMNDMC,
     $  ARSNKC,SRSNKC,
     $  ARDABA,SRDABA,
     $  ARDABB,SRDABB
C
CCCCCCC END OF BBN16-COMMON CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
        DATA PRCSR/1/,AP/1/
        DATA RL/0/,CMPLX/1/,FXD/2/,CNTG/1/,LNG/0/,SHRT/1/
        DATA IBUS1/64/,IBUS2/128/,IBUS3/192/
        DATA IFRMSZ/216/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C INITIALIZE MAP BUFFERS:
C
C USE BID 63 TO INITIALIZE NON-SNAP BUFFERS
C
        IDUM = 63
C
C
C NISTO <= ZEROS
C
        CALL MPCLB(IBUS1+IDUM,ANISTO,SNISTO,RL,CNTG,LNG)
        CALL VCLR(IDUM)
C
C
C TADBC <= APPROX 240 HZ SQUARE WAVE
C               (APPROX 30 BUFFERS/SEC. * 8 CYCLES/BUFFER => 240 HZ)
C       (MPWDB CAN'T WRITE DIRECTLY TO RL,SHRT BUFFER.
C        TADBC IS RL,SHRT, SO WRITE INITIAL DATA TO
C        THAN1, WHICH IS RL,LNG, & SAME #SAMPLES AS TADBC,
C        AND DO VMOVN FROM THAN1 TO TADBC.
```



```
C        THAN1 IS INITIALIZED TO ITS CORRECT VALUE LATER
C        IN THIS MODULE.)
C
        AMP = .055
        NCYCLE = 8
        CALL SQWAV(HWORK,IFRMSZ,NCYCLE,AMP)
        CALL MPWDB(THAN1,HWORK,4,1,HWORK(IFRMSZ))
C
        CALL MPCLB(IBUS1+IDUM,ATADBC,STADBC,RL,CNTG,SHRT)
        CALL VMOVN(IDUM,THAN1)
C
C
C TONES <= 1.
C       (TONES IS CONSTANT VECTOR:  SAMPLE INCREMENT=0)
C        SO ONLY HAVE TO INITIALIZE ONE SAMPLE)
C
        CALL MPTSB(TONES,1,1)
C
C
C TSP1 <= ZEROS
C
        CALL VCLR(TSP1)
C
C
C THAN0 <= HANNING WINDOW COEFFS
C
        CALL HANNNG(HWORK,265)
        CALL MPWDB(THAN0,HWORK,4,1,HWORK(265))
C
C
C TWSPZ <= ZEROS
C
        CALL VCLR(TWSPZ)
C
C
C TCOST <= COSINE TABLE
C
        CALL VCOST(TCOST,256.)
C
C
C TPSP1 <= ZEROS
C
        CALL VCLR(TPSP1)
C
C
C TE1 <= ZEROS
C
        CALL VCLR(TE1)
C
C
C THAN1 <= HANNING WINDOW COEFFS
```

```
C
      CALL HANNNG(NWORK,IFRMSZ)
      CALL MPWOB(THAN1,NWORK,4,1,NWORK(IFRMSZ))
C
C
C TLMU <= LAMBDA*MU(I)
C
      NWORK(1) = .035 *  .375
      NWORK(2) = .035 * -.25
      NWORK(3) = .035 *  .0625
      CALL MPWOB(TLMU,NWORK,4,1,NWORK(3))
C
C
C TAMR <= ONES
C
      CALL VMOV(TAMR,TONES)
C
C
C TFAC <= FACTORS FOR NOISE SHAPING ANALYSIS
C
      NWORK(1) = .684120
      NWORK(2) = .468032
      NWORK(3) = .320194
      NWORK(4) = .219054
      NWORK(5) = .149861
      NWORK(6) = .102524
      CALL MPWOB(TFAC,NWORK,4,1,NWORK(6))
C
C
C TQ1 <= ZEROS
C
      CALL VCLR(TQ1)
C
C
C TVN <= ZEROS
C
      CALL VCLR(TVN)
C
C
C TRN <= ZEROS
C
      CALL VCLR(TRN)
C
C
C TSNKA <= ZEROS
C
      CALL VCLR(TSNKA)
C
C
C TSNKB <= ZEROS
C
      CALL VCLR(TSNKB)
```



```
C
C
C TBTC <= BITSTREAM EQUIV OF SILENCE
C       (SET CODED ENERGY=0: $00NC=>12)
C
      CALL MPCLB(IBUS1+IDUM,ATBTC,STBTC,FXD,CNTG,LNG)
      DO 20 I=1,IFIX(STBTC)
 20   IWORK(I) = 12
      CALL MPWOB(IDUM,IWORK,2,1,IWORK(IFIX(STBTC)))
C
C
C TMUNA <= SILENCE W/ SYNC BIT = 0
C
      CALL MPCLB(IBUS1+IDUM,ATMUNA,STMUNA,FXD,CNTG,LNG)
      DO 30 I=1,IFIX(STMUNA)
 30   IWORK(I) = 12
      CALL MPWOB(IDUM,IWORK,2,1,IWORK(IFIX(STMUNA)))
C
C
C TMUNB <= SILENCE W/ SYNC BIT = 1
C
      CALL MPCLB(IBUS1+IDUM,ATMUNB,STMUNB,FXD,CNTG,LNG)
      DO 40 I=1,IFIX(STMUNB)
 40   IWORK(I) = 12
      IWORK(1) = IWORK(1) + 1
      CALL MPWOB(IDUM,IWORK,2,1,IWORK(IFIX(STMUNB)))
C
C
C RSRA <= ZEROS
C
      CALL VCLR(RSRA)
C
C
C RSRB <= ZEROS
C
      CALL VCLR(RSRB)
C
C
C RKN <= ZEROS
C
      CALL VCLR(RKN)
C
C
C RVHM <= ZEROS
C
      CALL VCLR(RVHM)
C
C
C RRN <= ZEROS
C
      CALL VCLR(RRN)
```

```
C
C
C RSNKC <= ZEROS
C
        CALL MPCLB(IBUS1+IDUM,ARSNKC,SRSNKC,RL,CNTG,SHRT)
        CALL VCLR(IDUM)
C
C
C RDANA <= ZEROS
C
        CALL MPCLB(IBUS1+IDUM,ARDANA,SRDANA,RL,CNTG,SHRT)
        CALL VCLR(IDUM)
C
C
C RDANB <= ZEROS
C
        CALL MPCLB(IBUS1+IDUM,ARDANB,SRDANB,RL,CNTG,SHRT)
        CALL VCLR(IDUM)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C INITIALIZE REAL SCALARS:
C
C
C
C TALPH <= PREEMPHASIS FILTER COEFFICIENT
C
        HTALPH = 0.4
        CALL MPWST(TALPH,HTALPH,1,1)
C
C
C TPWST <= ZERO
C
        CALL MPTSS(TPWST,0)
C
C
C TPSZI <= NEGATIVE RECIPROCAL OF PITCH EXTRACT. BUFFER SIZE
C
        HTPSZI = -1./265.
        CALL MPWST(TPSZI,HTPSZI,1,1)
C
C
C T14 <= 14.
C
        HT14 = 14.
        CALL MPWST(T14,HT14,1,1)
C
C
C TKTHR <= REAL,NULQ6 THRESHOLD
```



```
C
        HTKTHR = .9995
        CALL MPWST(TKTHR,HTKTHR,1,1)
C
C
C T72R <= 1./72.
C
        HT72R = 1./72.
        CALL MPWST(T72R,HT72R,1,1)
C
C
C T3R <= 1./3.
C
        HT3R = 1./3.
        CALL MPWST(T3R,HT3R,1,1)
C
C
C RDCF0-3 <= DEEMPHASIS FILTER COEFFICIENTS
C
        HRDCF2 = -0.4
        CALL MPTSS(RDCF0,0)
        CALL MPTSS(RDCF1,0)
        CALL MPWST(RDCF2,HRDCF2,1,1)
        CALL MPTSS(RDCF3,0)
C
C
C RDMY0-3 <= ZEROS
C
        CALL MPTSS(RDMY0,0)
        CALL MPTSS(RDMY1,0)
        CALL MPTSS(RDMY2,0)
        CALL MPTSS(RDMY3,0)
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C INITIALIZE INTEGER SCALARS:
C
C
C
        CALL MPIST(TSRFA,0)
        CALL MPIST(TSRFB,0)
        CALL MPIST(TBTFA,0)
        CALL MPIST(TBTFB,0)
        CALL MPIST(RBTFA,0)
        CALL MPIST(RBTFB,0)
        CALL MPIST(RSNFA,0)
        CALL MPIST(RSNFB,0)
        CALL MPIST(RUN,  0)
        CALL MPIST(ADPO, 0)
```

```
      CALL MPIST(TSNPO,-2)
      CALL MPIST(TBTPO,0)
      CALL MPIST(TNPO, 0)
      CALL MPIST(RNPO, 0)
      CALL MPIST(RNTPO,-2)
      CALL MPIST(RSNPO,0)
      CALL MPIST(DAPO, 0)
      CALL MPIST(TADDC,0)
      CALL MPIST(TNFC,0)
      CALL MPIST(RNFDC,0)
      CALL MPIST(RSNNR,0)
      CALL MPIST(TFCTR,0)
      CALL MPIST(NFCTR,0)
      CALL MPIST(RLCTR,0)
      CALL MPIST(RONNK,0)
      CALL MPIST(RSYNC,0)
      CALL MPIST(ADCTR,0)
      CALL MPIST(RNCTR,0)
      CALL MPIST(TNCTR,0)
      CALL MPIST(DACTR,0)
      CALL MPIST(TADPK,0)
      CALL MPIST(RNCOR,0)
      CALL MPIST(RERRS,0)
C
C
C
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE SQWAV(OUTBUF,ISIZE,NCYCLE,AMP)
C
C     PURPOSE
C             COMPUTES A SQUARE WAVE OF AMPLITUDE 'AMP',
C             WITH 'NCYCLE' CYCLES OCCURRING IN 'ISIZE' SAMPLES.
C
C     PARAMETERS
C             OUTBUF - (REAL) OUTPUT ARRAY
C             ISIZE - (INTEGER) NUMBER OF OUTPUT POINTS
C                     (MUST BE A MULTIPLE OF 'NCYCLE')
C             NCYCLE - (INTEGER) NUMBER OF SQUARE WAVE CYCLES
C             AMP - (REAL) SQUARE WAVE AMPLITUDE
C
C
      SUBROUTINE SQWAV(OUTBUF,ISIZE,NCYCLE,AMP)
C
      REAL OUTBUF(1)
C
      IBLKSZ = ISIZE/NCYCLE
C
      DO 10 I=0,NCYCLE-1
```



```
C
      DO 20 J=1,IBLKSZ/2
   20 OUTBUF((I * IBLKSZ) + J) = AMP
C
      DO 30 J=(IBLKSZ/2)+1,IBLKSZ
   30 OUTBUF((I * IBLKSZ) + J) = -AMP
C
   10 CONTINUE
C
C
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SUBROUTINE HAMMNG(OUTBUF,ISIZE)
C
C     PURPOSE
C             GENERATES HAMMING WINDOW OF SIZE 'ISIZE' (INTEGER)
C             INTO REAL ARRAY 'OUTBUF'.
C
C
      SUBROUTINE HAMMNG(OUTBUF,ISIZE)
C
      REAL OUTBUF(1)
C
      DANG = 3.1415926535 * 2./ISIZE
      DO 10 I=1,ISIZE
      OUTBUF(I) = .54 - .46*COS(DANG*(I-1))
   10 CONTINUE
C
      RETURN
      END
```

```
C  <DCA16>BBN16R.FOR.1    Mon 10-Nov-80 6:42PM          Page 1

C [BBND]<DCA16>BBN16R.FOR.1, 10-Nov-80 10:42:35, Ed: KFIELD
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                        BBN16R
C
C BOOT SEGMENT FOR BBN 16K BPS REAL-TIME SPEECH CODING SYSTEM
C
C DEFINES FORTRAN COMMON BLOCKS;
C OPENS MAP;
C CONFIGURES MAP BUFFERS AND SCALARS (BBN16C);
C INITIALIZES MAP BUFFERS AND SCALARS (BBN16I);
C PREBINDS FUNCTIONS AND DEFINES FUNCTION LISTS (BBN16F);
C EXECUTES SPEECH CODER (BBN16E);
C CLOSES MAP.
C
C COMMUNICATION BETWEEN CALLED SUBROUTINES IS VIA
C MAP-300 MEMORY, WHICH CONTAINS SYSTEM BUFFERS,
C SCALARS AND FUNCTION LISTS, AND VIA FORTRAN COMMON
C BLOCKS "BSIDS" (BUFFER AND SCALAR ID NAME DEFINITIONS),
C "BAS" (BUFFER ADDRESSES AND SIZES NEEDED BY BBN16I), AND
C "FLS" (FUNCTION LIST NAME DEFINITIONS).
C
C KFIELD 7/80
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C MAP FUNCTION LIST NAMES
C
        INTEGER BBNRTS,BBNLP,BBNFFT,BBNTIM,BBNRST
C
C
C [BBND]<MAP>BBN16-COMMON.FOR.12, 31-Oct-80 23:17:03, Ed: WOLF
C
C
C MAP BUFFER NAMES:
C
        INTEGER DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3
        INTEGER DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU
        INTEGER DVLUF
        INTEGER CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3
        INTEGER CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU
        INTEGER HISTO
        INTEGER TADBA,TADBB,TADBC,TSRA, TSRB, TONES
        INTEGER TSP0, TSP1, TSP,  THAMB,TWSP, TWSPZ
        INTEGER TCOST,TKSP, TKSPR,TKSPI,TPSP, TPSPR
        INTEGER TPSPI,TRP,  TC,   TWORK,TIC,  TCH
        INTEGER TE10, TE11, TE1,  THAM1,TWE1, TRS
        INTEGER TLMU, TRSP, TKE,  TKH,  TIK,  TFILT
        INTEGER TAH,  TAHR, TE2,  TFAC, TANS, TGFAC
        INTEGER TIDG, TUO,  TQ1,  TVH,  TRH0, TRH
```

```
C  <DCA16>BBN16R.FOR.1    Mon 10-Nov-80 6:42PM          Page 1:1

        INTEGER TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB
        INTEGER TBTC, TMDMA,TMDMB,TRPP
        INTEGER RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSRA
        INTEGER RSRB, RUHA, RUHB, RDGH, RCH,  RKH
        INTEGER RWH,  RVHM, RVH,  RRH0, RRH,  RRHDA
        INTEGER RRHDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB
C
C
C MAP REAL SCALAR NAMES:
C
        INTEGER TALPH,TPHST,TDCN, TFSZI,TMH,  TIM
        INTEGER T14,  TKTHR,TIG,  TC2P, T72R, T3M
        INTEGER RGH,  RMH,  RDCF0,RDCF1,RDCF2,RDCF3
        INTEGER RDNY0,RDNY1,RDNY2,RDNY3
C
C
C MAP INTEGER SCALAR NAMES:
C
        INTEGER TSRFA,TSRFB,TBTFA,TBTFB,TINC, TFENG
        INTEGER RBTFA,RBTFB,RSRFA,RSRFB,RUN
        INTEGER ADPO, TSRPO,TBTPO,TMPO, RMPO, RBTPO
        INTEGER RSRPO,DAPO, TADDC,TMFFC,RMFDC,RSNRR
        INTEGER TFCTR,RFCTR,RLCTR,RUNHK,RSYNC,RBOFU
        INTEGER ADCTR,TMCTR,RMCTR,DACTR,TAUPK
        INTEGER RNCOR,RERRS
C
C
C COMMON BLOCK - BUFFER AND SCALAR NAMES:
C
        COMMON /BSIDS/
     $  DVLC1,DVLC2,DVLC3,DVLK1,DVLK2,DVLK3,
     $  DVLK4,DVLK5,DVLK6,DVLG, DVLDG,DVLU,
     $  DVLUF,
     $  CTHC1,CTHC2,CTHC3,CTHK1,CTHK2,CTHK3,
     $  CTHK4,CTHK5,CTHK6,CTHG, CTHDG,CTHU,
     $  HISTO,
     $  TADBA,TADBB,TADBC,TSRA, TSRB, TONES,
     $  TSP0, TSP1, TSP,  THAMB,TWSP, TWSPZ,
     $  TCOST,TKSP, TKSPR,TKSPI,TPSP, TPSPR,
     $  TPSPI,TRP,  TC,   TWORK,TIC,  TCH,
     $  TE10, TE11, TE1,  THAM1,TWE1, TRS,
     $  TLMU, TRSP, TKE,  TKH,  TIK,  TFILT,
     $  TAH,  TAHR, TE2,  TFAC, TANS, TGFAC,
     $  TIDG, TUO,  TQ1,  TVH,  TRH0, TRH,
     $  TIUA, TIUB, TSNKA,TSNKB,TBTA, TBTB,
     $  TBTC, TMDMA,TMDMB,TRPP,
     $  RMDMA,RMDMB,RMDMC,RBTA, RBTB, RSRA,
     $  RSRB, RUHA, RUHB, RDGH, RCH,  RKH,
     $  RWH,  RVHM, RVH,  RRH0, RRH,  RRHDA,
     $  RRHDB,RSNKA,RSNKB,RSNKC,RDABA,RDABB,
     $  TALPH,TPHST,TDCN, TFSZI,TMH,  TIM,
```

```
     ) T14,  TKTMM,TIG,  TC2P, T72M, T3M,
     ) RGM,  RMM,  RDCF0,RDCF1,RDCF2,RDCF3,
     ) RDMY0,RDMY1,RDMY2,RDMY3,
     ) TSMFA,TSRFM,TMTFA,TMTFM,TINC, TFENG,
     ) RDTFA,RDTFB,RSMFA,RSMFB,RUB,
     ) ADPO, TSMPO,TBTPO,TMPO, RMPO, RMTPO,
     ) RSMPO,DAPO, TADDC,TMFFC,RMFDC,RSMNM,
     ) TFCTR,MFCTR,MLCTR,KOMMK,MSYNC,MBUFO,
     ) ADCTR,TNCTR,RNCTR,DACTR,TADPK,
     ) RNCOM,RERRS
C
C
C COMMON BLOCK - BUFFER ADDRESSES AND SIZES
C        NEEDED FOR INIT OF NON-SNAP BUFFERS
C        AND FOR FILE-TO-FILE MODE OF OPERATION
C
         COMMON /BAS/
     ) AMISTO,SMISTO,
     ) ATADBA,STADBA,
     ) ATADBB,STADBB,
     ) ATADBC,STADBC,
     ) ATBTC, STBTC,
     ) ATMDMA,STMDMA,
     ) ATMDMB,STMDMB,
     ) ARNDMA,SRNDMA,
     ) ARNDMB,SRNDMB,
     ) ARNDMC,SRNDMC,
     ) ANSMKC,SNSMKC,
     ) ARDABA,SRDABA,
     ) ARDABB,SRDABB
C
CCCCCCC END OF BBN16-COMMON CCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C COMMON BLOCK - FUNCTION LIST NAMES
C
         COMMON /FLS/
     ) BBNNTS,BBNLP,BBNPFT,BBNTIM,BBNRST
C
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C OPEN MAP, INHIBIT BUFFER CHECKING
C SET SNAP ERROR REPORT LEVEL:
C        0 => NO REPORT
C        1 => REPORT CODE; CONTINUE
C        2 => REPORT CODE; HALT
C        3 => REPORT MSG.; CONTINUE
C        4 => REPORT MSG.; HALT
C        5 => REPORT CODE; STATUS; HALT
```



```
C        6 => REPORT MSG.; STATUS; HALT
C
         IENPRT = 3
         CALL MPOPN(IENPRT)
         CALL MPINC(0)
C
C
C TYPE SYSTEM IDENTIFICATION
C
         TYPE 2
 2       FORMAT(' BBN 16K BPS SPEECH CODING SYSTEM'/)
C
C
C SET UP AND EXECUTE SYSTEM
C
         TYPE 10
 10      FORMAT(' CONFIGURING MAP BUFFERS & SCALARS...')
         CALL BBN16C
C
         TYPE 20
 20      FORMAT(' INITIALIZING MAP BUFFERS & SCALARS...')
         CALL BBN16I
C
         TYPE 30
 30      FORMAT(' PREBINDING FUNCTIONS & DEFINING FUNCTION LISTS...')
         CALL BBN16F
C
         TYPE 40
 40      FORMAT(' EXECUTING BBN16 SPEECH CODER...')
         CALL BBN16E
C
         CALL MPCLS(0)
C
C
C
         STOP
         END
```

```
C  <MAP>HIST16.FOR.7   Tue 30-Dec-80 9:27AM          Page 1

C [BBND]<MAP>HIST16.FOR.7, 30-Dec-80 09:27:25, Ed: WOLF
C    HIST16 -- PRINT OUT HISTOGRAM DATA FROM DCA16 SPEECH CODER

      INTEGER HISTO,FXD,CNTG,LNG,BUS1
      INTEGER H(260),G(64),DG(4),C1(8),C2(16),C3(8)
      INTEGER K1(64),K2(32),K3(16),K4(16),K5(16),K6(16)
C  EQUIVALENCE THE INDIVIDUAL PARAMETER ARRAYS TO THE BIG ARRAY H().
      EQUIVALENCE (H(1),G(1)),(H(65),DG(1)),(H(69),C1(1)),
     1 (H(77),C2(1)),(H(93),C3(1)),(H(101),K1(1)),(H(165),K2(1)),
     2 (H(197),K3(1)),(H(213),K4(1)),(H(229),K5(1)),(H(245),K6(1))
      DATA HISTO,BUS1,FXD,CNTG,LNG/1,64,2,1,0/
      DATA AHISTO,SHISTO/39944.,260./

      CALL MPOPN(3)
      TYPE 1
1     FORMAT(' FILE (OR TT:) FOR HISTOGRAM: '$)
      CALL ASSIGN(2,H(1),-1)
      CALL MPCLB(BUS1+HISTO,AHISTO,SHISTO,FXD,CNTG,LNG)
      CALL MPRUN(HISTO,H(1),2,0,H(260))
      NFR=0
      DO 5 I=1,4
5       NFR=NFR+DG(I)
      WRITE(2,10) NFR
10    FORMAT(' TOTAL FRAMES = ',I6//,'  N     G    DG    C1    C2',
     1 '    C3    K1    K2    K3    K4    K5    K6'/)
      DO 15 I=1,4
        J=I-1
15      WRITE(2,16) J,G(I),DG(I),C1(I),C2(I),C3(I),K1(I),K2(I),
        1 K3(I),K4(I),K5(I),K6(I)
16      FORMAT(I3,11I6)
      DO 20 I=5,8
        J=I-1
20      WRITE(2,21) J,G(I),C1(I),C2(I),C3(I),K1(I),K2(I),
        1 K3(I),K4(I),K5(I),K6(I)
21      FORMAT(I3,I6,6X,9I6)
      DO 25 I=9,16
        J=I-1
25      WRITE(2,26) J,G(I),C2(I),K1(I),K2(I),
        1 K3(I),K4(I),K5(I),K6(I)
26      FORMAT(I3,I6,6X,6X,I6,6X,7I6)
      DO 30 I=17,32
        J=I-1
30      WRITE(2,31) J,G(I),K1(I),K2(I)
31      FORMAT(I3,I6,4(6X),2I6)
      DO 35 I=33,64
        J=I-1
35      WRITE(2,36) J,G(I),K1(I)
36      FORMAT(I3,I6,4(6X),I6)
      WRITE(2,40)
40    FORMAT(/)
      CALL CLOSE(2)
      STOP
      END
```

DATE
ILMED
-8